CAR-TR-893
CS-TR-3923

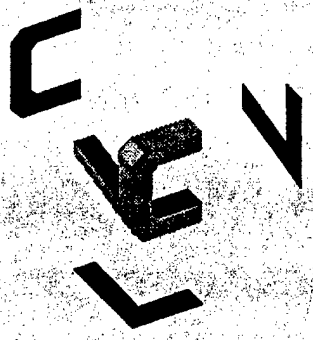
N00014-95-1-0521
August 1998

Extracting Structure from Optical Flow Using the Fast Error Search Technique

Sridhar Srinivasan

Center for Automation Research
University of Maryland
College Park, MD 20742-3275

COMPUTER VISION LABORATORY



CENTER FOR AUTOMATION RESEARCH

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742-3275

CAR-TR-893 N00014-95-1-0521
CS-TR-3923 August 1998

Extracting Structure from Optical Flow Using the Fast Error Search Technique

Sridhar Srinivasan

Center for Automation Research
University of Maryland
College Park, MD 20742-3275

Abstract

In this paper, we present a robust and computationally efficient technique for estimating the focus of expansion (FOE) of an optical flow field, using *fast partial search*. For each candidate location on a discrete sampling of the image area, we generate a linear system of equations for determining the remaining unknowns, *viz.* rotation and inverse depth. We compute the least squares error of the system without actually solving the equations, to generate an error surface that describes the goodness of fit across the hypotheses. Using Fourier techniques, we prove that given an $N \times N$ flow field, the FOE can be estimated in $\mathcal{O}(N^2 \log N)$ operations. Since the resulting system is linear, bounded perturbations in the data lead to bounded errors.

We support the theoretical development and proof of our algorithm with experiments on synthetic and real data. Through a series of experiments on synthetic data, we prove the correctness, robustness and operating envelope of our algorithm. We demonstrate the utility of our technique by applying it to the problem areas of 3D stabilization, moving object detection, rangefinding, obstacle detection, and generation of 3D models from video.

Keywords: Fast Partial Search, Focus of Expansion and Optical Flow

The support of this research by the Office of Naval Research under Grant N00014-95-1-0521 is gratefully acknowledged.

1 Introduction

The extraction of the three-dimensional structure of a moving scene from a sequence of images is termed the *structure from motion (SFM)* problem. The solution to this problem is a key step in the computer vision tasks of robotic navigation, obstacle avoidance, time to collision, recognition of solid objects, and surveillance. Recent interest in this area has been sparked by the desire to build 3D models from video for virtual reality, video conferencing, manufacturing and medical applications. The proliferation of powerful computing machinery and video streams has transitioned the problem from a purely theoretical domain to one of practical interest. Mathematical analysis of SFM proves the nonlinear interdependence of structure and motion given observations on the image plane. While this problem has received considerable attention by researchers, the proposed solutions tend to have several shortcomings. Yet, it is well recognized that the human visual system performs the task of simultaneously estimating motion and the qualitative depth structure of a moving scene quite efficiently and accurately.

Psychophysical experiments on human subjects reveal that the first stage of processing in the human visual system is the estimation of the motion field. Methods for estimating the structure from the motion field are referred to as *differential* techniques, as opposed to *discrete* methods which rely on two or more distinct views of the scene. The *optical flow*, which is defined as the 2-D projection of the 3D motion field, is composed of the horizontal and vertical velocity fields, $u(x,y)$ and $v(x,y)$. These are related to the motion and scene depth according to

$$\begin{aligned} u(x,y) &= (-t_x + xt_z)g(x,y) + xy\omega_x - (1+x^2)\omega_y + y\omega_z \\ v(x,y) &= (-t_y + yt_z)g(x,y) + (1+y^2)\omega_x - xy\omega_y - x\omega_z \end{aligned} \tag{1}$$

where the translational and rotational motion vectors are (t_x, t_y, t_z) and $(\omega_x, \omega_y, \omega_z)$ respectively. $g(x,y) = 1/Z(x,y)$ is the inverse scene depth, and all linear dimensions are normalized in terms of the focal length f of the camera. Such a pair of equations exists for every point (x,y) on the image plane at which optical flow can be determined. Assuming there exist M such points, there are $2M$ equations and $M+5$ independently determinable unknowns. Of the latter, M unknowns are related to scene depth and the remaining five are the determinable motion parameters. One obvious means of simplifying the system is to eliminate the depth map $g(x,y)$ from the above equations. Such an operation gives rise to a nonlinear system of equations in five determinable unknowns. In practice, however, the elimination of depth is very sensitive to noise in the data u, v, which is compounded by the nonlinear nature of the resulting equations.

The pioneering psychophysical experiments by Wallach and O'Connell [1] and Gibson [2, 3] hypothesize the simultaneous recoverability of structure and depth from an optical flow field. This is quantified and substantiated by Ullman [4], Nakayama and Loomis [6], and Koenderink and van Doorn [5]. The early approaches by Longuet-Higgins and Prazdny [7], and later by Waxman and Ullman [8] derive closed-form solutions to the SFM problem by examining derivatives of the optical flow field, with an underlying assumption of smoothness of the 3D surface. Bruss and Horn [9] and Adiv [10] propose nonlinear solutions which minimize the squared error between the observed and predicted flows. While both the approaches are iterative, the former is global while the latter subdivides

the flow field into smooth patches with a mechanism that allows them to be combined. Linear approaches to SFM can be traced back to the two-view solution formulated by Tsai and Huang [11] based on point correspondences. Linear methods are applied to motion fields by Zhuang *et al.* [12, 13], Waxman *et al.*, [14], and Mitiche *et al.* [15]. Prazdny [16] subdivides the SFM problem by first hypothesizing the rotational component of motion, and then solving for the remaining variables by means of a nonlinear system of equations. In a related manner, Lawton [17] searches for the translation direction at discrete points on the surface of a unit sphere he terms the *translation direction sphere.* A good summary of past literature and algorithms is provided in the books by Mitiche [18] and Weng *et al.* [19].

Among the more recent SFM literature are the solutions due to Jepson and Heeger [20] and Gupta and Kanal [21], both of which eliminate depth and rotation variables from (1) to solve for translation using a linear constraint. Direct techniques, first proposed by Aloimonos and Brown [22], and later expanded by Negahdaripour and Horn [23] and Horn and Welden [24], altogether bypass computation of the motion field by operating on raw luminance data. Several additional cues have been used to formulate a solution to the SFM problem, prominent among which are stereo images, stereo flow fields, and long sequences. These are not pertinent to the current work and are not discussed further. One important constraint that has gone largely unnoticed (expect as a means of resolving ambiguities in the form of multiple solutions) is the non-negativity of depth. This has been used in the recent approaches in a direct manner by Fermüller and Aloimonos [25, 26], and more indirectly by Fejes and Davis [27].

The stability of algorithms that eliminate the depth field $g(x, y)$ from (1) [11, 12, 13, 14, 15, 21] is questionable, although Gupta and Kanal [21] have claimed experimental results demonstrating some amount of noise resilience. Moreover, these algorithms discard valuable information in the form of nonlinear equality constraints which are not simple to enforce. Likewise, assuming smoothness of the depth field or optical flow field is not always valid, more so when there is noise in the flow estimates. Thus, differentiating noisy flow fields [7, 8] in order to solve SFM is highly undesirable. On the other hand, the method due to Fejes and Davis [27] requires strong variations in the underlying depth map (and thereby in the optical flow field) to work at all! Nonlinear optimization-based solutions [9, 10, 23] are relatively stable in the presence of noise. However, minimizing a nonlinear cost function exposes the solution to the pitfalls of local minima and slow convergence. The drawback of search-based methods [16, 17] is their slow speed of execution.

In this paper, we present a fast partial search technique for locating the focus of expansion (FOE) from a motion field. The focus of expansion is hypothesized to lie within a bounded square on the image plane. For each candidate location on a discrete sampling of the plane, we generate a linear system of equations for estimating the remaining unknowns which are the rotational velocities and inverse depth field. We compute the least squares error of the system *without actually solving the equations*, to generate an *error surface* that describes the goodness of fit as a function of the hypothesized focus of expansion. Our technique exploits the symmetry of (1) and uses Fourier techniques to vastly reduce the computational burden of this method based on partial search. The minimum of the error surface occurs at a discrete location very close to the true FOE. For

an image of size $N \times N$ in which the discrete sampling is at integer pixel locations and the FOE is assumed to occur within the image area, the order of complexity in computing the error surface is only $\mathcal{O}(N^2 \log N)$. Our method does not try to eliminate the depth field from (1). Since the resulting system is linear, bounded perturbations in the optical flow estimates lead to a deterministic, bounded offset of the error surface minimum from zero. Noise resilience is inherent in this linear formulation.

A significant deviation of our work from conventional techniques is that the error surface we generate gives us a distributed representation of the confidence, quantified in terms of the mean squared error over the solution space. In a similar manner, Simoncelli argues for a distributed representation of the optical flow field of a sequence as an alternative to a unique solution [28]. He shows that the computed error of a candidate solution is related to its likelihood of being the true solution. In a Bayesian framework, a distributed representation allows for error covariance propagation. Likewise, we claim that the error surface embeds information regarding both the presence of the true FOE and a confidence measure. A relatively flat error surface indicates low confidence in the solution whereas a sharp dip indicates high confidence.

In order to demonstrate the superiority of our approach over existing techniques, we use four criteria, *viz.* correctness, robustness, operating range and utility. At a minimum, any valid solution to a problem must be correct, given a perfect input data set. This is our first criterion, which we prove first in theory and then demonstrate on artificially generated perfect flow fields. Next, we show the robustness of our method by experiments on noisy and sparse flows. In order to traverse the operating range of the algorithm, we evaluate its performance over a range of fields of view and depth map structures. Finally, we consider several real-world applications calling for 3D SFM to demonstrate the utility of our approach. We use real imagery in this final set of experiments covering 3D stabilization, rangefinding, independent motion localization and obstacle detection.

This paper is organized as follows: Section 2 introduces the concept of partial search and the assumptions of our approach. The theory and derivations of our technique are explained in Section 3, with some of the proofs given in the Appendix. Section 4 extends our technique by relaxing certain assumptions made earlier. The experiments on synthetic and real data are detailed in Section 5. Finally, we conclude with a few remarks in Section 6.

2 Problem Formulation

In the remainder of this paper, we will assume a camera-centered coordinate system whose origin coincides with the center of the imaging lens and whose XY-plane is parallel to the image plane. We will also assume that the focal length of the lens is known accurately, and that the linear distance unit is normalized by the focal length, *i.e.* $f = 1$. The Z axis lies along the optical axis and the image plane lies on $Z = -1$, which is the focal plane of the camera. Equation (1) can be rewritten as

$$\begin{aligned} u(x,y) &= -(x - x_f)h(x,y) + xy\omega_x - (1 + x^2)\omega_y + y\omega_z \\ v(x,y) &= -(y - y_f)h(x,y) + (1 + y^2)\omega_x - xy\omega_y - x\omega_z \end{aligned} \tag{2}$$

where $(x_f, y_f) \stackrel{\text{def}}{=} (\frac{t_x}{t_z}, \frac{t_y}{t_z})$ is known as the *focus of expansion (FOE)* and $h(x,y)$ is the scaled inverse depth map of the scene being imaged, $h(x,y) = \frac{t_z}{Z(x,y)}$.

3D motion algorithms aim at computing the 3D motion parameters of the camera, *viz.* $t = (t_x, t_y, t_z)$ and $\omega = (\omega_x, \omega_y, \omega_z)$. It can be seen that t can only be recovered up to a scale factor, *i.e.* if $(t = t_0, \omega = \omega_0)$ is a solution, so is (mt_0, ω_0). The 3D structure of the scene $h(x,y)$ can be recovered, once the 3D motion parameters are known. Many techniques build on the first step in which $h(x,y)$ is eliminated from (2), but this makes the solution very sensitive to noise in flow estimates. In order to achieve noise resilience and yet keep the computational demands of the solution within reasonable bounds, we wish to devise a linear solution to the SFM problem. The technique we propose is based on fitting a linear model to each candidate hypothesis over a bounded search space.

2.1 Partial Search

Exhaustive search has seldom been accepted by theoreticians as the best technique for arriving at the solution of a set of equations, due to its perceived inelegance and lack of mathematical structure. However, in practice exhaustive search is used when other methods fail, and when it can be performed in a reasonable amount of time. Suppose the nonlinear set of equations for which a solution is desired is given by

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \qquad \mathbf{x} \in \Re^M, \mathbf{0} \in \Re^K, K \geq M \tag{3}$$

Exhaustive search for a solution $\mathbf{x}_e$ involves (i) enumerating a finite set of candidate solutions $\mathcal{X} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots\}$ that adequately cover the solution space, (ii) computing an error metric (*e.g.* $\|\mathbf{f}(\mathbf{x}_i)\|^2$) which associates each candidate solution $\mathbf{x}_i$ with a compliance measure, and (iii) locating the minimum error and corresponding candidate solution, which for the squared error metric is

$$\mathbf{x}_e = \arg\min_{\mathbf{x} \in \mathcal{X}} \{\|\mathbf{f}(\mathbf{x})\|^2\} \tag{4}$$

In general, the order of complexity of exhaustive search is proportional to the cardinality $|\mathcal{X}|$ of $\mathcal{X}$. This can get unmanageably large as the dimensionality M of $\mathbf{x}$ increases.

An intermediate approach to searching for all the components of the solution is to enumerate only a few components and solve for the remaining components based on the hypothesis. For example in (3), assume that the argument $\mathbf{x}$ can be partitioned as

$$\mathbf{x} = \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix}, \qquad \mathbf{a} \in \Re^{M_1}, \mathbf{b} \in \Re^{M_2}, M_1 + M_2 = M \tag{5}$$

and given $\mathbf{a}_i$, (3) can be solved for the remaining components $\mathbf{b}_i$ with a small number of computations. We will call $\mathbf{a}$ the *search component* and $\mathbf{b}$ the *dependent component* of $\mathbf{x}$. Partial search of a solution $\mathbf{x}_p$ is performed by (i) enumerating a finite set of candidate partial solutions $\mathcal{A} \in \{\mathbf{a}_0, \mathbf{a}_1, \ldots\}$ that adequately cover the search component space, (ii) computing the dependent component $\mathbf{b}_i$ corresponding to each $\mathbf{a}_i \in \mathcal{A}$ that satisfies (or closely satisfies) (3), (iii) computing an error metric, for example $\|\mathbf{f}([\mathbf{a}_i' | \mathbf{b}_i']')\|^2$ for the squared error case, and (iv) picking the candidate solution corresponding with the

minimum error. Step (ii) can be defined as a minimization over the continuous space of permissible $\mathbf{b}$ of the same squared error metric to formulate the partial search solution $\mathbf{x}_p$ as

$$\begin{aligned} \mathbf{x}_p &= \begin{pmatrix} \mathbf{a}^* \\ \mathbf{b}^* \end{pmatrix} \\ \mathbf{a}^* &= \arg\min_{\mathbf{a}\in\mathcal{A}} \min_{\mathbf{b}} \left\| \mathbf{f}\begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} \right\|^2 \\ \mathbf{b}^* &= \arg\min_{\mathbf{b}} \left\| \mathbf{f}\begin{pmatrix} \mathbf{a}^* \\ \mathbf{b} \end{pmatrix} \right\|^2 \end{aligned} \tag{6}$$

Partial search separates the problem into a search and a minimization. In general, the complexity of the original problem is proportional to the cardinality $|\mathcal{A}|$ of $\mathcal{A}$, and to the number of operations required to compute $\mathbf{b}_i$ given $\mathbf{a}_i$. It may be possible, depending on the problem, to simplify certain steps in the partial search procedure, leading to further reduction in complexity. This paper approaches the SFM problem through partial search and exploits the symmetry of the problem to minimize the computational requirement.

When the optical flow u, v is known at M points $\{(x_0, y_0), (x_1, y_1), \ldots (x_{M-1}, y_{M-1})\}$, we get M instances of (2) with $2M$ linearly independent equations[1] and $M+5$ unknowns in all. An exhaustive search for a solution must consider the combinatorial multiplicity of the $M+5$ unknowns. This gets unwieldy even when $M = 5$, which is the minimum number of equation sets needed for a unique solution. For the sake of argument, assuming that there are L possible discretized permissible values for each of the scalar variables, exhaustive search computation requires $\mathcal{O}(L^{M+5})$ operations. A crude search with 5% uncertainty in each variable (*i.e.* $L = 10$) and with $M = 5$ observed points gives rise to 10^{10} combinations, which escalates to over 10^{15} when the number of observations doubles to 10. Exhaustive search is clearly not an alternative, and indeed we require that the number of search combinations be independent of the number of observations since the latter can potentially be huge.

In order to restrict the search to a finite number of dimensions, the search component $\mathbf{a}$ must not include the depth variables $h(x_i, y_i)$. Of the remaining five variables, the rotational components are linearly related to the observation, which in this case is the flow field. Thus, in addition to the scaled inverse depth map, rotation forms an ideal candidate for including in the dependent component. The search component comprises x_f and y_f which are the focus of expansion, and the dependent component includes the rotation and depth variables which number $M+3$ in all. It can be seen in (2) that the dependent component and observation are linearly related when the search component or FOE is fixed. Given a search component $\mathbf{a}_i = (x_{f_i}, y_{f_i})'$, the dependent component $\mathbf{b}_i$ can be solved for by means of a $(M+3) \times (M+3)$ matrix inversion which takes at most $\mathcal{O}(M^3)$ steps. The worst-case computational requirement of this partial search approach is $\mathcal{O}(M^3 L^2)$, which is combinatorially smaller than $\mathcal{O}(L^{M+5})$. Yet even this number is too large for practical use. The remainder of this section analyzes the linear

[1]Except for the singular case where there is no rotation and one of the M points coincides with the FOE.

system obtained when choosing a candidate search component, with the eventual aim of reducing the order of complexity of partial search.

2.2 Assumptions

We will begin the analysis by enumerating the assumptions which simplify the understanding of our approach. As we progress, we will loosen some of the assumptions until only the essential ones remain. Further, we will argue that the assumptions are realistic and commonly hold in practice. In the following paragraphs describing the assumptions, the first sentence is the initial assumption made, for ease of understanding. The second sentence, which is *italicized*, is the minimum necessary to prove our method. The remainder of the item is a discussion of the assumption.

- The input flow field is defined over a square region, of size $N \times N$, where $N = 2^k$. *Can be fully relaxed.* With a square field whose side is a power of 2, we can use the fast Fourier transform (FFT) in some of our later computations. Other sizes do not change the formulation, but might lead to certain bounded inefficiencies in the computational requirement.

- The input flow field is dense. *Can be fully relaxed.* We first prove our result by assuming that a dense flow field is available for the $N \times N$ image. Later, we discuss the handling of sparse flow fields. However, the complexity of the proposed method is related to the size of the image for which the flow is available and not to the number of equation sets (*i.e.* the number of points at which the optical flow is known). Thus, the number of computations required to solve the problem given a sparse flow field is equal to the corresponding count given a dense flow field.

- The FOE lies within the area defined by the input flow field. *The FOE lies within a bounded region.* The primary aim of this effort is directed toward situations where there is a large forward translational motion, like vehicle navigation. In this case, the FOE almost always lies within the image. In extreme situations where the FOE is very large or at infinity, t_z is very small in comparison with one or both of t_x and t_y. Equation (1) can be approximated as

$$\begin{aligned} u(x,y) &\approx -t_x g(x,y) + xy\omega_x - (1+x^2)\omega_y + y\omega_z \\ v(x,y) &\approx -t_y g(x,y) + (1+y^2)\omega_x - xy\omega_y - x\omega_z \end{aligned} \tag{7}$$

  For this system of equations, there are only $M + 4$ independently determinable variables since there is still a scale ambiguity in the system. However, that is not all. Vision systems in practice tend to have small fields of view, or equivalently, large focal lengths. In this case, $x, y < 1$ and the contribution of the second terms in (7) is small, leading to a first-order approximation

$$\begin{aligned} u(x,y) &\approx -t_x g(x,y) - \omega_y + y\omega_z \\ v(x,y) &\approx -t_y g(x,y) + 1\omega_x - x\omega_z \end{aligned} \tag{8}$$

  The instability of system (7) is clear when one notes the occurrence of multiple solutions in its approximation (8). Assume that $\{t_x, t_y, \omega_x, \omega_y, \omega_z, g_0, g_1, \ldots g_{M-1}\}$

is a valid solution for the M instances of (8). Then the sets $\{t_x, t_y, \omega_x + t_y\delta, \omega_y - t_x\delta, \omega_z, g_0 + \delta, g_1 + \delta, \ldots g_{M-1} + \delta\}$ are also valid solutions for all values of $\delta > 0$. The positivity constraint on δ ensures that depths are non-negative. An analysis of the instability of SFM under various conditions is provided in Adiv [29] and Young and Chellappa [30].

Secondly, when performing 3D SFM over a long sequence of frames, the location of the FOE at the previous time instant is known after the first frame for which the motion field is recovered. If it can be assumed that the FOE does not change drastically between frames, the search window for the FOE at the current frame can be shifted and re-centered around the estimated FOE at the previous frame. Under the relaxed assumption that the FOE must lie within a bounded area, the re-centering strategy will work.

Finally, it must be borne in mind that the output of our algorithm is not a single solution; rather, it is an error metric for the entire search space. When the FOE lies outside the search space, we expect that the shape of the error surface will show a dip in the direction of the true FOE. In our experiments, we have observed that when the true FOE lies outside the search area, the minimum is attained at its periphery, indicating the likelihood of the true minimum lying beyond. This flags an error condition and provides a candidate offset by which to shift the search area.

In summary, the only non-trivial situation where we expect the proposed method to fail is when the following three conditions simultaneously hold: (i) wide field of view, (ii) very small forward translation, and (iii) no available estimate of the FOE from the prior frame that can form a reasonable guess to recenter the search area. Even in this situation, if the minimum error is attained at the search boundary (which is very likely to be the case), we can iteratively re-center and locate the error surface minimum. In the next section, we develop the theory behind our approach.

3 Approach

Let the true FOE be (x_f, y_f). Assuming that the flow field is of size $N \times N$ and all N^2 flow estimates are available, the optical flow at pixel location $i, j \in \{0, 1, \ldots, N-1\}^2$ is given by

$$\begin{aligned} u_{i,j} &= -(x_{i,j} - x_f)h_{i,j} + x_{i,j}y_{i,j}\omega_x - (1 + x_{i,j}^2)\omega_y + y_{i,j}\omega_z \\ v_{i,j} &= -(y_{i,j} - y_f)h_{i,j} + (1 + y_{i,j}^2)\omega_x - x_{i,j}y_{i,j}\omega_y - x_{i,j}\omega_z \end{aligned} \tag{9}$$

where $x_{i,j} = \frac{i-(N-1)/2}{f}$, $y_{i,j} = \frac{j-(N-1)/2}{f}$, and $\{h, u, v\}_{i,j} = \{h, u, v\}(x_{i,j}, y_{i,j})$. The transformation between the pixel coordinate system point (i, j) and the normalized 3D coordinate system point (x, y) on the image plane is reversible and is given by

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} xf + \frac{N-1}{2} \\ yf + \frac{N-1}{2} \end{pmatrix} \tag{10}$$

Thus, the optical center lies at $\left(\frac{N-1}{2}, \frac{N-1}{2}\right)$ in the pixel coordinate system. WLOG, we will switch between coordinate systems to simplify notation wherever necessary. Define

$$
\begin{aligned}
r_{i,j} &= \begin{pmatrix} x_{i,j}y_{i,j} & -(1+x_{i,j}^2) & y_{i,j} \end{pmatrix}' \\
s_{i,j} &= \begin{pmatrix} 1+y_{i,j}^2 & -x_{i,j}y_{i,j} & -x_{i,j} \end{pmatrix}' \\
\mathbf{Q} &= \begin{bmatrix} r'_{0,0} \\ s'_{0,0} \\ r'_{0,1} \\ s'_{0,1} \\ \vdots \\ r'_{N-1,N-1} \\ s'_{N-1,N-1} \end{bmatrix} \\
h &= -\begin{pmatrix} h_{0,0} & h_{0,1} & \dots & h_{N-1,N-1} \end{pmatrix}' \\
\mathbf{u} &= \begin{pmatrix} u_{0,0} & v_{0,0} & u_{0,1} & v_{0,1} & \dots & u_{N-1,N-1} & v_{N-1,N-1} \end{pmatrix}' \\
\mathbf{P}(x_f, y_f) &= \begin{bmatrix} x_{0,0}-x_f & 0 & 0 & & & \\ y_{0,0}-y_f & 0 & 0 & & & \\ 0 & x_{0,1}-x_f & 0 & & \mathbf{0} & \\ 0 & y_{0,1}-y_f & 0 & & & \\ 0 & 0 & x_{0,2}-x_f & & & \\ 0 & 0 & y_{0,2}-y_f & & & \\ & & & \ddots & \vdots & \\ & \mathbf{0} & & \dots & x_{N-1,N-1}-x_f & \\ & & & & y_{N-1,N-1}-y_f & \end{bmatrix} \\
\mathbf{A}(x_f, y_f) &= \begin{bmatrix} \mathbf{P}(x_f, y_f) & \mathbf{Q} \end{bmatrix} \\
\mathbf{x}_0 &= \begin{bmatrix} h \\ \omega \end{bmatrix}.
\end{aligned} \tag{11}
$$

We will drop the argument (x_f, y_f) of $\mathbf{P}(x_f, y_f)$ and $\mathbf{A}(x_f, y_f)$ where it is obvious. The above definitions allow us to consolidate the motion equations for all individual flow vectors in the brief form

$$\begin{bmatrix} \mathbf{P}(x_f, y_f) & \mathbf{Q} \end{bmatrix} \begin{bmatrix} h \\ \omega \end{bmatrix} = \mathbf{u} \tag{12}$$

i.e.

$$\mathbf{A}(x_f, y_f)\, \mathbf{x}_0 = \mathbf{u}. \tag{13}$$

Replacing the unknowns x_f, y_f and $\mathbf{x}_0$ by the hypothesized variables x_h, y_h and $\mathbf{x}$ we get the general condition

$$\mathbf{A}(x_h, y_h)\, \mathbf{x} \rightarrow \mathbf{u} \tag{14}$$

where the true solution exactly satisfies (14).

We now define a squared error cost function $C(x_h, y_h, \mathbf{x})$ as

$$C(x_h, y_h, \mathbf{x}) = \|\mathbf{A}(x_h, y_h)\, \mathbf{x}\|_2^2 \tag{15}$$

Since

$$C(x_h, y_h, \mathbf{x}) \geq 0 \tag{16}$$

$$C(x_f, y_f, \mathbf{x}_0) = 0 \tag{17}$$

(i) the true solution to the system (14) minimizes the cost function $C(\ )$, and (ii) all minimizers of $C(\ )$ satisfy (14) exactly. Thus, we have reduced the original problem to

$$\min_{x_h, y_h, \mathbf{x}} C(x_h, y_h, \mathbf{x}) \tag{18}$$

which can be decomposed as

$$\min_{x_h, y_h} \min_{\mathbf{x}} C(x_h, y_h, \mathbf{x}) \tag{19}$$

The inner minimization occurs at the least squares (LS) solution $\mathbf{x}_{LS}$ of $\mathbf{A}(x_h, y_h)\ \mathbf{x} \rightarrow \mathbf{u}$. Later, we will prove uniqueness of $\mathbf{x}_{LS}$. Indeed, $C(\ )$ can have more than one minimizer. In the separable form (19), existence of multiple minima is indicated by existence of multiple values of x_h, y_h (and thereby of $\mathbf{x}$) attaining the minimum for $C(\ )$. In this situation, exhaustive search over the entire search space will pick out all the valid solutions. However, it is not difficult to see that even with coarse discretization, the number of free variables is too large to permit exhaustive search. Referring to our earlier discussion of search, we see that partial search is an ideal technique for solving (19).

In order to perform partial search, we set $\{x_h, y_h\}$ to be the search component and $\mathbf{x}$ to be the dependent component. We discretize the search component space at midway locations between four pixels[2] over the entire image area, in line with our assumption that the FOE lies within the image. Later, we will relax the search component space to be any uniformly discretized rectangular set of points, not necessarily within the image. We do not dispute that one could potentially construct a pathological counter-example that lets the minima "slip through" the lattice formed by discrete values. But it is our belief that such situations do not occur in practice. We will experimentally justify the discretization process.

The LS solution $\mathbf{x}_{LS}$ of $\mathbf{A}(x_h, y_h)\ \mathbf{x} \rightarrow \mathbf{u}$ satisfies

$$\mathbf{A}'\mathbf{A}\mathbf{x}_{LS} = \mathbf{A}'\mathbf{u} \tag{20}$$

$$\begin{bmatrix} \mathbf{P}'\mathbf{P} & \mathbf{P}'\mathbf{Q} \\ \mathbf{Q}'\mathbf{P} & \mathbf{Q}'\mathbf{Q} \end{bmatrix} \mathbf{x}_{LS} = \begin{bmatrix} \mathbf{P}' \\ \mathbf{Q}' \end{bmatrix} \mathbf{u} \tag{21}$$

where the arguments of $\mathbf{A}$ and $\mathbf{P}$ have been dropped. $\mathbf{D} \stackrel{\text{def}}{=} \mathbf{P}'\mathbf{P}$ is a diagonal matrix, given by

$$\begin{aligned} \mathbf{D} &= \text{Diag}\left\{\bar{x}_i^2 + \bar{y}_i^2\right\} \\ \mathbf{D}^{-1} &= \text{Diag}\left\{\frac{1}{\bar{x}_i^2 + \bar{y}_i^2}\right\} \end{aligned} \tag{22}$$

where $\bar{x}_i$ and $\bar{y}_i$ are functions of (x_h, y_h):

$$\begin{aligned} \bar{x}_i &= x_{\lfloor i/N \rfloor, i \bmod N} - x_h \\ \bar{y}_i &= y_{\lfloor i/N \rfloor, i \bmod N} - y_h \end{aligned} \tag{23}$$

[2]Hypothesizing the FOE to lie at pixel locations leads to a singularity.

We shall assume here that $\bar{x}_i$ and $\bar{y}_i$ are never zero. This can be achieved by ensuring that the discretization grid for FOE hypotheses is not coincident with the pixel sampling grid. Even this restriction can be overcome, but we shall deal with it later, assuming for now that $\mathbf{D}$ is never singular. One way of guaranteeing this is to pick the hypothesized FOE to lie midway between pixels. When $\mathbf{D}$ is nonsingular, $\mathbf{x}_{LS}$ is unique. Applying appropriate pre-multiplying matrices, we can manipulate the system as shown:

$$\begin{aligned}
\begin{bmatrix} \mathbf{D} & \mathbf{P}'\mathbf{Q} \\ \mathbf{Q}'\mathbf{P} & \mathbf{Q}'\mathbf{Q} \end{bmatrix} \mathbf{x} &= \begin{bmatrix} \mathbf{P}' \\ \mathbf{Q}' \end{bmatrix} \mathbf{u} \\
\Rightarrow \begin{bmatrix} \mathbf{I} & \mathbf{D}^{-1}\mathbf{P}'\mathbf{Q} \\ \mathbf{Q}'\mathbf{P} & \mathbf{Q}'\mathbf{Q} \end{bmatrix} \mathbf{x} &= \begin{bmatrix} \mathbf{D}^{-1}\mathbf{P}' \\ \mathbf{Q}' \end{bmatrix} \mathbf{u} \\
\Rightarrow \begin{bmatrix} \mathbf{I} & \mathbf{D}^{-1}\mathbf{P}'\mathbf{Q} \\ \mathbf{0} & \mathbf{Q}'\mathbf{Q} - \mathbf{Q}'\mathbf{P}\mathbf{D}^{-1}\mathbf{P}'\mathbf{Q} \end{bmatrix} \mathbf{x} &= \begin{bmatrix} \mathbf{D}^{-1}\mathbf{P}' \\ \mathbf{Q}'(\mathbf{I} - \mathbf{P}\mathbf{D}^{-1}\mathbf{P}') \end{bmatrix} \mathbf{u}
\end{aligned} \tag{24}$$

Introducing matrices $\mathbf{M} \in \Re^{2N \times 2N}$ and $\hat{\mathbf{M}} \in \Re^{3\times 3}$ defined as

$$\begin{aligned}
\mathbf{M} &\stackrel{\text{def}}{=} (\mathbf{I} - \mathbf{P}\mathbf{D}^{-1}\mathbf{P}') \\
&= \text{BlockDiag}\left\{ \frac{1}{\bar{x}_i^2 + \bar{y}_i^2} \begin{bmatrix} \bar{y}_i^2 & -\bar{x}_i\bar{y}_i \\ -\bar{x}_i\bar{y}_i & \bar{x}_i^2 \end{bmatrix} \right\} \qquad (25) \\
\hat{\mathbf{M}} &\stackrel{\text{def}}{=} \mathbf{Q}'\mathbf{M}\mathbf{Q} \qquad (26)
\end{aligned}$$

we get

$$\begin{aligned}
\begin{bmatrix} \mathbf{I} & \mathbf{D}^{-1}\mathbf{P}'\mathbf{Q} \\ \mathbf{0} & \hat{\mathbf{M}} \end{bmatrix} \mathbf{x} &= \begin{bmatrix} \mathbf{D}^{-1}\mathbf{P}' \\ \mathbf{Q}'\mathbf{M} \end{bmatrix} \mathbf{u} \\
\Rightarrow \begin{bmatrix} \mathbf{I} & \mathbf{D}'\mathbf{P}'\mathbf{Q} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{x} &= \begin{bmatrix} \mathbf{D}^{-1}\mathbf{P}' \\ \hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M} \end{bmatrix} \mathbf{u} \\
\Rightarrow \mathbf{x} &= \begin{bmatrix} \mathbf{D}^{-1}\mathbf{P}'(\mathbf{I} - \mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}) \\ \hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M} \end{bmatrix} \mathbf{u}
\end{aligned} \tag{27}$$

The error $\mathbf{A}\mathbf{x} - \mathbf{u}$ is

$$\begin{aligned}
\mathbf{A}\mathbf{x} - \mathbf{u} &= \begin{bmatrix} \mathbf{P} & \mathbf{Q} \end{bmatrix} \begin{bmatrix} \mathbf{D}^{-1}\mathbf{P}'(\mathbf{I} - \mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}) \\ \hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M} \end{bmatrix} \mathbf{u} - \mathbf{u} \\
&= \left(\mathbf{M}\mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M} - \mathbf{M} \right) \mathbf{u}
\end{aligned} \tag{28}$$

giving the squared error

$$\begin{aligned}
\|\mathbf{A}\mathbf{x} - \mathbf{u}\|^2 &= \mathbf{u}' \left(\mathbf{M}\mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M} - \mathbf{M} \right)^2 \mathbf{u} \\
&= \mathbf{u}'\mathbf{M}^2\mathbf{u} + \mathbf{u}'\mathbf{M}\mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}^2\mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}\mathbf{u} \\
&\quad - 2\mathbf{u}'\mathbf{M}^2\mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}\mathbf{u}
\end{aligned} \tag{29}$$

Again, note that $\mathbf{A}, \mathbf{M}$ and $\hat{\mathbf{M}}$ are functions of (x_h, y_h). It can be easily verified that $\mathbf{M}$ is idempotent, *i.e.* $\mathbf{M}^2 = \mathbf{M}$. (29) simplifies to

$$\|\mathbf{Ax} - \mathbf{u}\|^2 = \mathbf{u}'\mathbf{Mu} - \mathbf{u}'\mathbf{M}\mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{Mu} \tag{30}$$

The most naive strategy for computing the least squared error, or equivalently, of performing the inner minimization in (19), is to explicitly solve the linear system for the unknown $\mathbf{x}$ and use this estimate to evaluate the squared error. Without assuming any sparseness or symmetry in the coefficient matrix $\mathbf{A}$, the system can be solved by matrix inversion. Keeping in mind that $\mathbf{x} \in \Re^{N^2+3}$, the order of complexity of estimating $\mathbf{x}$ in this manner is $\mathcal{O}(N^6)$[3]. Exploiting the structure of $\mathbf{A}$ leads to dramatic improvements. From (25) and (26), it can be seen that computing matrices $\mathbf{M}$ and $\hat{\mathbf{M}}$ requires $\mathcal{O}(N^2)$ operations, which is the complexity of estimating $\mathbf{x}$ and the squared error as well. Taking into account the outer minimization search leads to an overall complexity of $\mathcal{O}(N^8)$ for the matrix inversion method and $\mathcal{O}(N^4)$ by exploiting symmetry. In addition, examining (30) reveals that the only data-dependent term is $\mathbf{u}$. Thus, given sufficient memory, the data-independent terms can be pre-computed and stored, for each value of (x_h, y_h). However, even with this strategy, the overall complexity cannot be brought down below $\mathcal{O}(N^4)$.

In what forms the core of this effort, we will show that the structure in $\mathbf{A}$ can be further exploited so that the errors can be computed *directly*, without computing the solution $\mathbf{x}$ explicitly. Moreover, the least squared errors for all the candidate hypotheses can be computed in a single step, which leads to an overall complexity of $\mathcal{O}(N^2 \log N)$. At first, this seems ridiculous since factoring out N^2 from the complexity introduced by the outer search leaves $\mathcal{O}(\log N)$ which is insufficient even for vector addition. Yet, it is the simultaneous estimation of all errors in the search space that allows such a low overall complexity. We introduce the notion of *Fast Computability* in the following section.

3.1 Fast Computability

A few preliminary definitions and theorems are necessary before we proceed with the proof of our technique. Proofs of the Fast Computability theorems are given in the Appendix.

Definition 1 *Let* $\mathbf{S} \in \Re^{N \times N}$. *The* cyclic shift $\mathbf{S}[i_0, j_0]$ *of* $\mathbf{S}$ *by* (i_0, j_0) *is defined by*

$$\mathbf{S}[i_0, j_0]_{i,j} = \mathbf{S}_{(i+i_0) \bmod N, (j+j_0) \bmod N} \tag{31}$$

Definition 2 *The lexical ordering of a matrix* $\mathbf{S} \in \Re^{N \times N}$, *denoted by* $\mathcal{S} \in \Re^{N^2 \times N^2}$, *is defined by*

$$\mathcal{S}_{i,j} = [\mathcal{L}(\mathbf{S})]_{i,j} = \begin{cases} 0 & i \neq j \\ \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N} & i = j \end{cases} \tag{32}$$

$\mathcal{L}$ *is the lexical ordering operator. The inverse operation is defined only on diagonal matrices:*

$$[\mathcal{L}^{-1}(Diag(s_k))]_{i,j} = s_{iN+j} \tag{33}$$

[3]More accurately, $\mathcal{O}(N^{2 \log_2 7})$ [31]!

We denote the space of such diagonal matrices by $\mathcal{L}^{N^2}(\Re)$, where the argument $\Re$ denotes the space of each element s_i of the matrix $\mathcal{S}$.

Definition 3 *The cyclic shift of a lexically ordered matrix $\mathcal{S} \in \mathcal{L}^{N^2}(\Re)$, by (i_0, j_0), is*

$$\mathcal{S}[i_0, j_0] = \mathcal{L}\left(\mathcal{L}^{-1}(\mathcal{S}[i_0, j_0])\right) = \mathcal{L}\left(\mathbf{S}[i_0, j_0]\right) \tag{34}$$

Definition 4 *The quantity $q(i_0, j_0)$ is said to be* Fast Computable (FC) *if $q(i_0, j_0)$ can be evaluated $\forall i_0, j_0 \in \{0, 1, \ldots, N-1\}$ in $O(N^2 \log N)$ computational steps.*

Theorem 1 *Let $a, b \in \Re^{N^2}$ and $\mathcal{S} \in \mathcal{L}^{N^2}(\Re)$. The quantity $q(i_0, j_0) \in \Re$ defined by $q(i_0, j_0) = a'\mathcal{S}[i_0, j_0]b$ is* FC.

Proof: Appendix A.1.

We now extend Theorem 1 to a more complicated situation where each scalar element of the above data structures, including matrices and vectors, is replaced by a *doublet.* The doublet corresponding to a scalar matrix entry is defined as a 2×2 submatrix and a doublet of a vector component as a 2-vector. In other words, each scalar element in the matrices is replaced by a 2×2 real matrix, and each scalar element in the vectors by a 2×1 real vector. The concepts of cyclic shift, lexical ordering and inverse lexical ordering are redefined below.

Definition 5 *Let*

$$\begin{aligned}
\mathbf{S}_{i,j} &= \begin{pmatrix} \mathbf{S}_{i,j_{00}} & \mathbf{S}_{i,j_{01}} \\ \mathbf{S}_{i,j_{10}} & \mathbf{S}_{i,j_{11}} \end{pmatrix} \in \Re^{2\times 2} \\
\mathbf{S} &= \begin{bmatrix} \mathbf{S}_{0,0} & \mathbf{S}_{0,1} & & \mathbf{S}_{0,N-1} \\ \mathbf{S}_{1,0} & \mathbf{S}_{1,1} & \cdots & \mathbf{S}_{1,N-1} \\ & \vdots & \ddots & \\ \mathbf{S}_{N-1,0} & \mathbf{S}_{N-1,1} & & \mathbf{S}_{N-1,N-1} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{S}_{0,0_{00}} & \mathbf{S}_{0,0_{01}} & \mathbf{S}_{0,1_{00}} & & \mathbf{S}_{0,N-1_{01}} \\ \mathbf{S}_{0,0_{10}} & \mathbf{S}_{0,0_{11}} & \mathbf{S}_{0,1_{10}} & \cdots & \mathbf{S}_{0,N-1_{11}} \\ \mathbf{S}_{1,0_{00}} & \mathbf{S}_{1,0_{01}} & \mathbf{S}_{1,1_{00}} & & \mathbf{S}_{1,N-1_{01}} \\ & \vdots & & \ddots & \\ \mathbf{S}_{N-1,0_{10}} & \mathbf{S}_{N-1,0_{11}} & \mathbf{S}_{N-1,1_{10}} & & \mathbf{S}_{N-1,N-1_{11}} \end{bmatrix} \in \Re^{2N\times 2N}
\end{aligned} \tag{35}$$

The cyclic shift $\mathbf{S}[i_0, j_0]$ of $\mathbf{S}$ by (i_0, j_0) is defined by

$$\begin{aligned}
\mathbf{S}[i_0, j_0]_{i,j} &= \mathbf{S}_{(i+i_0)\bmod N,(j+j_0)\bmod N} \\
i.e. \qquad \mathbf{S}[i_0, j_0] &= \begin{bmatrix} \mathbf{S}_{i_0 \bmod N, j_0 \bmod N} & \cdots & \mathbf{S}_{i_0 \bmod N,(N-1+j_0)\bmod N} \\ \vdots & \ddots & \\ \mathbf{S}_{(N-1+i_0)\bmod N, j_0 \bmod N} & & \mathbf{S}_{(N-1+i_0)\bmod N,(N-1+j_0)\bmod N} \end{bmatrix}
\end{aligned} \tag{36}$$

Definition 6 *The lexical ordering of the doublet matrix* $\mathbf{S}$ *shown in (35), denoted by* $\mathcal{S} \in \Re^{2N^2 \times 2N^2}$, *is defined by*

$$\begin{aligned} \mathcal{S} &= BlockDiag\left\{\mathbf{S}_{\lfloor i/N \rfloor, i \bmod N}\right\} \\ &= BlockDiag\left\{\begin{pmatrix} \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{00}} & \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{01}} \\ \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{10}} & \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{11}} \end{pmatrix}\right\} \end{aligned} \tag{37}$$

In keeping with Definition 2, we will denote the space of permissible $\mathcal{S}$ *matrices which constitute the lexical ordering of a doublet matrix by* $\mathcal{L}^{N^2}(\Re^{2\times 2})$.

Definition 7 *The cyclic shift of a lexically ordered matrix* $\mathcal{S} \in \mathcal{L}^{N^2}(\Re^{2\times 2})$ *is defined by (34).*

Theorem 2 *Let* $\mathcal{S} \in \mathcal{L}^{N^2}(\Re^{2\times 2})$, *and* $a, b \in \Re^{2N^2}$. *The quantity* $q(i_0, j_0) = a'\mathcal{S}[i_0, j_0]b$ *is FC.*

Proof: Appendix A.2.

3.2 Non-cyclic Shift

The last step in setting the stage to prove fast computability of the squared error is extending Theorem 2 to non-cyclic shifts. When a space-limited data sequence is shifted across a viewing window, points that were undefined earlier appear within the window. In a cyclic shift, the data points of the original sequence that disappear from the viewing area are wrapped around to fill the locations within the newly visible area. However, when the shift is not cyclic, it is necessary to define its behavior, especially with regard to how emerging areas are filled in.

Definition 8 *The non-cyclic shift* $\mathbf{S}\langle i_0, j_0\rangle$ *of matrix* $\mathbf{S} \in \Re^{N\times N}$ *by* (i_0, j_0) *is defined in terms of the* superset matrix $\bar{\mathbf{S}} \in \Re^{2N\times 2N}$ *by*

$$\mathbf{S}\langle i_0, j_0\rangle_{i,j} = \bar{\mathbf{S}}_{i+i_0, j+j_0}, \qquad \forall i_0, j_0 \in \{0, 1, \ldots, N-1\}. \tag{38}$$

Definition 9 *The non-cyclic shift of a lexically ordered matrix* $\mathcal{S} \in \mathcal{L}^{N^2}(\Re)$ *is defined by*

$$\mathcal{S}[i_0, j_0] = \mathcal{L}\left(\mathbf{S}\langle i_0, j_0\rangle\right) \tag{39}$$

Theorem 3 *Let* $a, b \in \Re^{N^2}$ *and* $\mathcal{S} \in \mathcal{L}^{N^2}(\Re)$. *The quantity* $q(i_0, j_0) \in \Re$ *defined by* $q(i_0, j_0) = a'\mathcal{S}\langle i_0, j_0\rangle b$ *is FC.*

Proof: Appendix A.3.

We now extend Theorem 3 to the doublet space.

Theorem 4 *Let* $a, b \in \Re^{2N^2}$ *and* $\mathcal{S} \in \mathcal{L}^{N^2}(\Re^{2\times 2})$. *The quantity* $q(i_0, j_0) \in \Re$ *defined by* $q(i_0, j_0) = a'\mathcal{S}\langle i_0, j_0\rangle b$ *is FC.*

Proof: Appendix A.4.

Finally, we extend the above theorem to arbitrary finite-dimensional aggregations in the following results.

Corollary 1 *Let $\mathcal{S} \in \mathcal{L}^{N^2}(\Re^{2\times 2})$, $a \in \Re^{2N^2}$ and $B \in \Re^{2N^2 \times p}$ for some arbitrary integer $p \geq 1$. The quantity $q(i_0, j_0) = a'\mathcal{S}\langle i_0, j_0\rangle B$ is FC.*

Proof: Writing out $q(i_0, j_0)$ in terms of its components,

$$q(i_0, j_0) = \left(\begin{array}{cccc} a'\mathcal{S}[i_0, j_0]b_0 & a'\mathcal{S}[i_0, j_0]b_1 & \ldots & a'\mathcal{S}[i_0, j_0]b_{p-1} \end{array} \right) \tag{40}$$

where $B = (b_0 \;\; b_1 \;\; \ldots \;\; b_{p-1})$. Each of the components of $q(i_0, y_0)$ is FC. When p is a constant, the overall computational steps are still $O(N^2 \log N)$.

Corollary 2 *Likewise, let $A, B \in \Re^{2N^2 \times p}$, p being an arbitrary constant. $Q(i_0, j_0) = A'\mathcal{S}[i_0, j_0]B$ is FC.*

Proof: As in Corollary 1.

3.3 Basic Proof

Next, we show how the squared error given by equation (30) is Fast Computable, for every choice of (x_f, y_f). This enables the likely solution space to be searched exhaustively in $\mathcal{O}(N^2 \log N)$ steps. First, we use (10) to define the pixel coordinate (i_h, j_h) corresponding to the hypothesized FOE (x_h, y_h) as

$$\left(\begin{array}{c} i_h \\ j_h \end{array} \right) = \left(\begin{array}{c} fx_h + \frac{N-1}{2} \\ fy_h + \frac{N-1}{2} \end{array} \right) \tag{41}$$

We define the search space for (i_h, j_h) to be $\left\{\frac{1}{2}, \frac{3}{2}, \ldots, N - \frac{1}{2}\right\}^2$. The candidate solutions lie at half-pixel displacements along a regular grid covering the image area. In the discussion ahead, we will interchangably use the pixel and XY coordinate systems.

Lemma 1 $\mathbf{M} \in \mathcal{L}^{N^2}(\Re^{2\times 2})$

Proof: From (25) and Definition 6.

Definition 10 *The superset matrix $\bar{\mathbf{M}} \in \Re^{4N\times 4N}$ is a doublet matrix defined by*

$$\bar{\mathbf{M}}_{i,j} = \frac{\left[\begin{array}{cc} (j - N + \frac{1}{2})^2 & -(i - N + \frac{1}{2})(j - N + \frac{1}{2}) \\ -(i - N + \frac{1}{2})(j - N + \frac{1}{2}) & (i - N + \frac{1}{2})^2 \end{array} \right]}{(i - N + \frac{1}{2})^2 + (j - N + \frac{1}{2})^2} \tag{42}$$

$\forall i, j \in \{0, 1, \ldots, 2N - 1\}$.

In the above definition, the $\frac{1}{2}$ terms are included to stagger the grid. There is a one-to-one relation between the hypothesized location (i_h, j_h) and the index (i, j) according to

$$\{i_h, j_h\} \leftrightarrow \{i, j\} + \frac{1}{2} \tag{43}$$

The importance of staggering the sampling grid for the search component is seen from (42), where the $\frac{1}{2}$ terms prevent the denominator of the leading fraction from vanishing. However, it must be noted that this fix is merely cosmetic since even without staggering, the entry $\bar{\mathbf{M}}_{N,N}$ at the singular point can be evaluated using limits.

Lemma 2 $\mathbf{M}(x_h, y_h) = \mathbf{M}\langle N-1-\lfloor i_h \rfloor, N-1-\lfloor j_h \rfloor\rangle$ *to within half-pixel discretization.*

Proof: From (23), (25) and (10), we have

$$\begin{aligned}
&\mathcal{L}^{-1}(\mathbf{M}(x_h, y_h))_{i,j} \\
&= \frac{\begin{bmatrix} (y_{i,j}-y_h)^2 & -(x_{i,j}-x_h)(y_{i,j}-y_h) \\ -(x_{i,j}-x_h)(y_{i,j}-y_h) & (x_{i,j}-x_h)^2 \end{bmatrix}}{(x_{i,j}-x_h)^2 + (y_{i,j}-y_h)^2} \\
&= \frac{\begin{bmatrix} (j-\frac{N-1}{2}-fy_h)^2 & -(i-\frac{N-1}{2}-fx_h)(j-\frac{N-1}{2}-fy_h) \\ -(i-\frac{N-1}{2}-fx_h)(j-\frac{N-1}{2}-fy_h) & (i-\frac{N-1}{2}-fx_h)^2 \end{bmatrix}}{(i-\frac{N-1}{2}-fx_h)^2 + (j-\frac{N-1}{2}-fy_h)^2}
\end{aligned} \tag{44}$$

Comparing (42) with (44), we see that $\mathcal{L}^{-1}(\mathbf{M})$ is a windowed version of the superset matrix $\hat{\mathbf{M}}$. The location of this window (k, l) is computed by equating the indices. In general, since the computed location may not be integral, rounding is performed. The following equation evaluates k. l is evaluated likewise.

$$\begin{aligned}
k - N + \frac{1}{2} &\rightarrow i - \frac{N-1}{2} - fx_h \\
k &= \lceil i + N - fx_h - \frac{N+1}{2} \rceil \\
&= i + (N-1) - \lfloor fx_h + \frac{N-1}{2} \rfloor \\
&= i + (N-1) - \lfloor i_h \rfloor
\end{aligned} \tag{45}$$

Thus, upon discretization, we get

$$\mathcal{L}^{-1}(\mathbf{M}(x_h, y_h))_{i,j} = \bar{\mathbf{M}}_{i+(N-1)-\lfloor i_h \rfloor, j+(N-1)-\lfloor j_h \rfloor} \tag{46}$$

which gives the non-cyclic shift relationship in mixed coordinate system notation

$$\mathbf{M}(x_h, y_h) = \mathbf{M}\langle N-1-\lfloor i_h \rfloor, N-1-\lfloor j_h \rfloor\rangle \tag{47}$$

This result is key to tying in fast computation with the SFM problem. Lemma 2 maps the FOE search space $\left[-\frac{N}{2}, \frac{N}{2}\right]^2$ to the discrete non-cyclic shifts $i_0, j_0 \in \{N-1, N-2, \ldots, 1, 0\}$ of $\mathbf{M}\langle i_0, j_0\rangle$.

Lemma 3 $\hat{\mathbf{M}}$ *is FC.*

Proof:

$$\hat{\mathbf{M}} = \mathbf{Q}'\mathbf{M}(x_f, y_f)\mathbf{Q} = \mathbf{Q}'\mathbf{M}\langle i_0, j_0\rangle\mathbf{Q} \tag{48}$$

which is FC, from Corollary 2 and Lemma 2, since $\mathbf{Q} \in \Re^{2N^2 \times 3}$.

Lemma 4 $\hat{\mathbf{M}}^{-1}$ *is FC.*

Proof: From Lemma 3, $\hat{\mathbf{M}}$ can be evaluated over the search space in $\mathcal{O}(N^2 \log N)$ operations. A 3×3 matrix can be inverted in a constant number p of operations. Inverting $\hat{\mathbf{M}}$ over all the N^2 hypotheses takes N^2p operations. The overall complexity is still $\mathcal{O}(N^2 \log N)$ and hence $\hat{\mathbf{M}}^{-1}$ is FC.

Lemma 5 $\mathbf{u}'\mathbf{M}\mathbf{u}$ *and* $\mathbf{Q}'\mathbf{M}\mathbf{u}$ *are FC.*

Proof: From Corollary 1 and Corollary 2.

Theorem 5 *The squared error (30) is FC.*

Proof: Evaluating (30) in the manner indicated by the underbraces in the equation

$$\|\mathbf{A}\mathbf{x} - \mathbf{u}\|^2 = \underbrace{\mathbf{u}'\mathbf{M}\mathbf{u}} - \underbrace{\mathbf{u}'\mathbf{M}\mathbf{Q}}\,\hat{\mathbf{M}}^{-1}\,\underbrace{\mathbf{Q}'\mathbf{M}\mathbf{u}} \tag{49}$$

groups the right-hand side as a sum or product of Fast Computable terms. The products are among 3×3 matrices and 3×1 vectors. The product and sum operations themselves require $O(N^2)$ computations to evaluate over the entire search space. The overall complexity is $O(N^2 \log N)$, and the squared error is FC.

In (49), it can be seen that the terms $\mathbf{u}'\mathbf{M}\mathbf{Q}$ and $\hat{\mathbf{M}} = \mathbf{Q}'\mathbf{M}\mathbf{u}$ are transposes of each other. There are, in all, three quantities whose evaluation is based on the Fast Computation theorems, *viz.* $\mathbf{u}'\mathbf{M}\mathbf{u}$, $\mathbf{u}'\mathbf{M}\mathbf{Q}$ and $\mathbf{Q}'\mathbf{M}\mathbf{Q}$. Of these, the last term is data-independent. In our basic solution, therefore, $\hat{\mathbf{M}}$ and its inverse can be computed beforehand. Likewise, the Fourier transform of the superset matrix $\bar{\mathbf{M}}$, which is evaluated for computing $\hat{\mathbf{M}}$, is stored in memory for later use in computing $\mathbf{u}'\mathbf{M}\mathbf{u}$ and $\mathbf{u}'\mathbf{M}\mathbf{Q}$. Next, we list the steps of the basic algorithm.

3.4 Algorithm

- *Step 1:* Compute the superset matrix $\bar{\mathbf{M}}$ and its Fourier transform.
- *Step 2:* Compute $\hat{\mathbf{M}}\langle i_0, j_0\rangle = \mathbf{Q}'\mathbf{M}\langle i_0, j_0\rangle\mathbf{Q}$ and its inverse for all locations of the candidate solution.
- *Step 3:* Using the given dense optical flow field, compute $\mathbf{u}'\mathbf{M}\langle i_0, j_0\rangle\mathbf{u}$ and $\mathbf{u}'\mathbf{M}\langle i_0, j_0\rangle\mathbf{Q}$.
- *Step 4:* Form the products and compute the error for each candidate hypothesis of the FOE.

- *Step 5:* Pick the location of the FOE corresponding to the smallest squared error.
- *Step 6:* Repeat from step 3 for the next data set.

The basic building block of the algorithm is a function that performs the Fast Computation $a'\mathbf{M}\langle i_0, j_0\rangle b$ for the arguments a and b. Since this is central to the approach, we have provided a detailed algorithm for this function, which we term *FastCompute*, in Appendix B.

4 Relaxing the Assumptions

How does the proposed solution change if the velocity estimate u, v is not available for a particular point (i, j) ? In such a situation the corresponding equation pair (9) is also not available. If, with no increase in computational complexity, we can replace (9) with a set of equations that retains the coefficients of $h_{i,j}$ and yet does not influence the solution in any manner, we can extend our technique to both sparse flow fields as well as non-2^k image sizes.

Consider the equations

$$\begin{aligned} 0 &= -(x_{i,j} - x_f)h_{i,j} + 0\omega_x + 0\omega_y + 0\omega_z \\ 0 &= -(y_{i,j} - y_f)h_{i,j} + 0\omega_x + 0\omega_y + 0\omega_z \end{aligned} \tag{50}$$

The consistent solution of these equations is $h_{i,j} = 0$, with ω being indeterminate. Since the coefficients of ω are zero, appending this set of equations to our system does not influence the solution. Obviously, the depth at point i, j cannot be estimated. Replacing (9) by (50) for every pixel at which the optical flow is not known preserves the structure of $\mathbf{P}(x_f, y_f)$, although the corresponding rows of $\mathbf{Q}$ must be set to zero. With this substitution, the reasoning in Section 3 holds and the FOE can still be estimated in $\mathcal{O}(N^2 \log N)$ steps.

The above substitution allows us to zero-pad a flow field if needed to make its size a power of 2. Moreover, since most optical flow techniques produce sparse flow fields, this substitution allows us to interface with these methods. There is no need to interpolate sparse flow fields — an operation which may lead to reduced accuracy. Yet another situation calling for zero-padding is where the search area is larger than the image, even when the latter is a power of 2 in size.

Indeed, (50) is a special case of weighting (9) by zero. By using a continuum of weights, reliability measures of the flow field can be incorporated into the system. The depth map must be suitably rescaled to retain the coefficients of $h_{i,j}$ so as to preserve the structure of $\mathbf{P}(x_f, y_f)$ and thereby validate the reasoning.

If it is known by some means that the FOE is present in an area around $x_{\text{off}}, y_{\text{off}}$, our method can be made more effective by incorporating this knowledge into the formulation. Assume that the offset in the pixel coordinate system is $i_{\text{off}}, j_{\text{off}}$ corresponding

to $x_{\text{off}}, y_{\text{off}}$. The superset matrix $\bar{\mathbf{M}}$ is redefined by

$$\bar{\mathbf{M}}_{i,j} = \frac{\begin{bmatrix} (j - j_{\text{off}} - N + \frac{1}{2})^2 & -(i - i_{\text{off}} - N + \frac{1}{2})(j - j_{\text{off}} - N + \frac{1}{2}) \\ -(i - i_{\text{off}} - N + \frac{1}{2})(j - j_{\text{off}} - N + \frac{1}{2}) & (i - i_{\text{off}} - N + \frac{1}{2})^2 \end{bmatrix}}{(i - i_{\text{off}} - N + \frac{1}{2})^2 + (j - j_{\text{off}} - N + \frac{1}{2})^2} \tag{51}$$

$\forall i, j \in \{0, 1, \ldots, 2N-1\}$. This definition offsets the search window by $i_{\text{off}}, j_{\text{off}}$. The remainder of the computation process remains the same. However, the offset must be added to $\arg\min_{x_h, y_h} \|\mathbf{A}(x_h, y_h)\mathbf{x} - \mathbf{u}\|^2$ when estimating the FOE.

Shifting the search area by an offset allows the fusion of external information into our algorithm. For example, while processing a sequence of images, the search window can be re-centered at the FOE estimate of the previous frame. If it is assumed that acceleration is small from frame to frame, such re-centering improves the probability of finding the true FOE within the search area. Alternatively, if a kinematic model exists for the camera platform, the velocities can be predicted from the past behavior. This provides a useful starting guess for the FOE.

Finally, we claim without proof that the overall complexity of estimating the FOE hypothesized to lie in a $M_x \times M_y$ area (potentially offset from the center), given a flow field of size $N_x \times N_y$ (potentially sparse), is $\mathcal{O}(M_2 N_2 \log M_2 N_2)$, where $M_2 = 2^{\lceil \log_2(M_x + N_x) \rceil}$ and $N_2 = 2^{\lceil \log_2(M_y + N_y) \rceil}$. Although this is within the same order of magnitude for images whose aspect ratio is bounded, this expression is derived by tightening the Fast Computability proofs.

5 Experiments and Results

In this section, we describe our experiments on evaluating the partial search FOE estimation algorithm. Our experiments comprise two phases, *viz.* a first phase which measures quantitative performance on synthetic data generated with known parameters, followed by a second phase which examines qualitative performance on real-world imagery with emphasis on useful applications. Using synthetic data allows us to accurately characterize performance over a range of situations. In the first phase, the optical flow field is synthesized. On real data, the flow is obtained from a sequence of images using optical flow techniques described later. The synthetic data experiments demonstrate the correctness, operating range and robustness of our algorithm. The utility of the algorithm in solving real-world problems is shown using real data.

During the course of the experiments, the proposed algorithm is applied to the flow field and the location corresponding to the minimum error in (30) is picked as the FOE $(\hat{x}, \hat{y})$ (or equivalently, $(\hat{i}, \hat{j})$ in the pixel coordinate system). A correction of $\frac{1}{2}$ pixel is applied to each direction, to undo the effect of staggering (43). Once the FOE is estimated, the angular velocity and depth map can be obtained by solving linear equations. From (27), we can see that the LS estimate for ω is

$$\hat{\omega} = \hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}\mathbf{u} \tag{52}$$

which is a product of terms that have already been calculated. Therefore, the estimate of rotation is available with no extra computation.

Set	N	f	i_f	j_f	ω_x	ω_y	ω_z	d
A	256	400	102.0	51.0	-2.0	5.0	8.0	1.5
B	256	400	127.5	201.5	5.0	3.0	4.0	1.7

Table 1: Parameters used in generating synthetic data sets A and B

The depth map takes some additional effort to recover. Using (27), we get

$$h = \mathbf{D}^{-1}\mathbf{P}'(\mathbf{I} - \mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M})\mathbf{u} \tag{53}$$

which requires the computation of $\mathbf{D}^{-1}\mathbf{P}'\mathbf{Q}$ and $\mathbf{D}^{-1}\mathbf{P}'\mathbf{u}$. These take $\mathcal{O}(N^2)$ operations. It must be noted that the depth map is very sensitive to perturbations in the optical flow as well as the FOE estimate, especially near the latter. Although depth map recovery is not the main emphasis of this work, we will give relevant experimental results.

5.1 Experiments with Synthetic Data

The least squared error as a function of the hypothesized FOE is denoted by E, and is referred to as the *error surface*. The minimum value of the error surface, which occurs at $(\hat{i}, \hat{j})$, is denoted by $E_{\min}$. Incorporating the round-off error in estimating the FOE, the error in the FOE estimate, ϵ, is given by

$$\epsilon = \min\left\{ \left\| \begin{matrix} \hat{i} - i_0 \\ \hat{j} - j_0 \end{matrix} \right\|, \left\| \begin{matrix} \hat{i} - i_0 \\ \hat{j} - j_1 \end{matrix} \right\|, \left\| \begin{matrix} \hat{i} - i_1 \\ \hat{j} - j_0 \end{matrix} \right\|, \left\| \begin{matrix} \hat{i} - i_1 \\ \hat{j} - j_1 \end{matrix} \right\| \right\} \tag{54}$$

where $\{i,j\}_0 = \lfloor \{i,j\}_f - \frac{1}{2} \rfloor + \frac{1}{2}$ and $\{i,j\}_1 = \lceil \{i,j\}_f - \frac{1}{2} \rceil + \frac{1}{2}$. The $\frac{1}{2}$ terms compensate for staggering of the hypothesis grid. This expression provides a 4-pixel neighborhood associated with zero error, if the true FOE lies off the $\frac{1}{2}$-pixel offset grid lines along each axis. When the true FOE is exactly on a grid point, the neighborhood shrinks to one pixel.

In many favorable situations, our algorithm estimates the FOE with no error. We use a secondary error metric, ϵ_ω, which is the total angular error, given by

$$\epsilon_\omega = \|\hat{\omega} - \omega\| \tag{55}$$

to characterize and differentiate between cases where ϵ is not usable.

5.1.1 Generation of Synthetic Data

Using (9), we generate optical flow fields corresponding to chosen inverse depth maps, FOEs and rotational velocities. The depth map, in the real world, is comprised of largely smooth regions bounded by sharp discontinuities. The size of these regions varies widely. In order to closely model the real world, the depth map is generated using a fractal model. This is motivated by the use of fractal models for luminance images which, like depth maps, are also composed of largely smooth regions and discontinuities. The Fourier transform method is used to generate the depth map fractal. In this method, the

Set	$\hat{i}$	$\hat{j}$	ϵ	$\hat{\omega}_x$	$\hat{\omega}_y$	$\hat{\omega}_z$	ϵ_ω
A	101.5	51.5	0.00	-1.985	5.013	-7.995	0.020
B	127.5	201.5	0.00	5.000	3.000	4.000	0.000

Table 2: Performance on dense, noiseless flow fields

transform magnitude is chosen to be exponential, of the form $(f_x^2 + f_y^2)^{-d/2}$. The phase is random, uniformly distributed in $[0, 2\pi)$. We call d the *fractal exponent.* Upon inverse Fourier transformation, the resulting images (both real and imaginary components of the transform) are fractals. Choosing smaller fractal exponents gives more intricate fractals. The fractals so generated are used as synthetic inverse depth maps.

For our experiments, we chose $N = 256$. We generated two baseline data sets A and B with parameters shown in Table 1. The angular velocities are in m rad, and the focal length f is in pixel units. The FOE is shown in the pixel coordinate system, with the origin located at the top left corner of the image frame. The i axis is down the rows and j axis across the columns. The FOE in *Set B* is located at a valid grid point while the FOE in *Set A* is located exactly midway between four grid points. Besides, *Set A* displays a relatively large rotation along the optical axis. On the other hand, the dominant rotation, for *Set B*, is along the image plane axes. In addition, the FOE coordinates are well distributed in $[0, N]$. The focal length of 400 pixels reflects a normal field of view of 35 degrees.

Fig. 1 depicts *Set A*: the underlying fractal depth map is shown as a grayscale image in (a), the rotational component of the flow field in (b), the translational component alone in (c), and the overall synthesized flow in (d). Close objects are bright while objects at infinity are black in Fig. 1(a). The confounding of rotational and translational flow is clear from (b),(c) and (d). In particular, there are areas in Fig. 1(d) where the flow field alternates direction. Such inflections occur at points where neither the rotation nor translation is dominant. *Set B* is similarly depicted in Figs. 2(a)-(d).

5.1.2 Sensitivity Analysis on Synthetic Data

Next, we look at the performance of our algorithm on the synthetic data sets. Sets A and B will serve as the benchmark. For analyzing the sensitivity of our algorithm to a particular parameter p, we generate an ensemble of flow fields using different values of p, keeping the remaining parameters fixed to those of A and B. To begin with, we assume that the focal length is known accurately. In Section 5.1.7, we examine the effect of errors in the focal length estimate.

5.1.3 Ideal Case

With a perfect flow field and accurate focal length estimate, our algorithm locates the FOE without error. The rotational error ϵ_ω is zero for *Set B* which lies on a valid grid point, but non-zero, though small, for *Set A*, which lies between grid points. These results are shown in Table 2. The rotational velocities and error ϵ_ω are in 10^{-3} rad. Contours of

Noise		*Sparsity*				
σ	η (approx.)	100%	80%	60%	40%	20%
0.0	0.00	0.00	0.00	0.00	0.00	0.00
0.1	2.65	0.00	0.00	0.00	0.00	0.00
0.2	5.29	0.00	0.00	0.00	0.00	0.50
0.4	10.51	1.00	1.00	1.21	1.12	0.50
1.0	25.08	5.41	9.80	3.55	6.77	6.62

Table 3: FOE estimation error ϵ as a function of the noise and sparsity of the input flow field

the error surface E are plotted in Figs. 3(a) and (b) for sets A and B respectively. The true location of the FOE is marked by a + and the estimated location by a $\times$.

5.1.4 Performance with Sparse/Noisy Flows

In the real world, optical flow is seldom determined at all points in an image with certainty. Besides, the flow estimates are typically noisy. The sparsity of flow depends on the specific optical flow algorithm chosen, the presence of local high-frequency information, and the existence of a coherent motion across the image. We simulate a sparse flow field by randomly including or discarding the flow at a given pixel, according to an *i.i.d.* binary distribution. We realize noise in the flow field by adding an *i.i.d.* zero-mean Gaussian process with variance σ to each component of velocity. The noise level η is measured according to the angular error metric employed in [32], given by

$$\eta = E\left[\cos^{-1}\left(\frac{\mathbf{v_0}\cdot\mathbf{v}}{\|\mathbf{v_0}\|\|\mathbf{v}\|}\right)\right] \tag{56}$$

where $\mathbf{v_0} = (u\ v\ 1)'$, $\mathbf{v} = \mathbf{v_0} + (\eta_u\ \eta_v\ 0)'$ and $\eta_u, \eta_v \sim \mathcal{N}(0,\sigma)$. η, measured here in degrees, is insensitive to the magnitude of the motion vector and offers a normalized measure against which a range of velocities can be compared meaningfully.

We simulated flow fields corresponding to five combinations of sparsity, *viz.* 100%, 80%, 60%, 40% and 20%, and five combinations of noise level, $\sigma = 0, 0.1, 0.2, 0.4$ and 1.0. The angular error η corresponding to σ is approximately the same for a given σ, over sets A and B and over all sparsity levels. The error ϵ in estimating the FOE using our algorithm is tabulated as a function of noise and sparsity in Table 3. Fig. 4 represents the *worst case* scenario. The flow fields generated with $\sigma = 1.0$ are shown in Fig. 4, (a) *Set A* with 80% density and (b) *Set B* with 20% density. Figs. 4(c) and (d) plot the error surface contours with the true FOE (+) and estimate ($\times$) corresponding to (a) and (b) respectively, showing good compliance of our solution.

5.1.5 Efect of Depth Structure

Early 3D SFM techniques that assume a smooth flow field tend to fail when there are discontinuities introduced by busy depth maps. Likewise, the newer techniques that exploit depth non-negativity minimally require busy depth maps to work at all. One of the

d	$\hat{i}$	$\hat{j}$	ϵ	$\hat{\omega}_x$	$\hat{\omega}_y$	$\hat{\omega}_z$	ϵ_ω
Set A							
Planar	102.5	51.5	0.0	-1.989	4.985	-8.002	0.019
1.7	101.5	50.5	0.0	-2.014	5.014	-8.009	0.022
1.5	101.5	50.5	0.0	SET A			
1.3	101.5	50.5	0.0	-2.012	5.015	-7.992	0.021
1.1	101.5	50.5	0.0	-2.009	5.014	-7.999	0.017
Set B							
All	127.5	201.5	0.0	5.000	3.000	4.000	0.000

Table 4: Performance as a function of depth structure quantified by the fractal exponent

Field of View		Noise Level		Estimates						
f	θ (degree)	η	σ	$\hat{i}$	$\hat{j}$	ϵ	$\hat{\omega}_x$	$\hat{\omega}_y$	$\hat{\omega}_z$	$\epsilon\omega$
Set A										
200	65.2	10.0	0.408	101.5	51.5	0.00	-1.935	5.060	-7.959	0.097
280	49.1	10.0	0.368	101.5	50.5	0.00	-2.018	5.009	-8.011	0.023
560	25.8	10.0	0.512	101.5	51.5	0.00	-1.985	5.007	-8.018	0.024
800	18.2	10.0	0.820	94.5	49.5	7.07	-2.003	5.055	-7.921	0.096
Set B										
200	65.2	10.0	0.372	127.5	201.5	0.00	5.010	3.002	3.996	0.011
280	49.1	10.0	0.345	127.5	201.5	0.00	5.007	3.011	3.968	0.035
560	25.8	10.0	0.580	125.5	199.5	2.83	4.990	3.016	4.026	0.032
800	18.2	10.0	0.910	130.5	205.5	5.00	5.022	2.984	3.987	0.030

Table 5: Performance as a function of field of view

merits of our approach is its applicability to both of these extremes and to intermediate cases. We validate this claim by studying performance on flow fields simulated using several depth maps generated by a range of fractal exponents d. Moreover, we also consider a planar depth map for comparison, given by $h_{\text{planar}}(i,j) = C_0 + C_x i + C_y j$. An imaged 3D planar scene gives rise to this form of depth map.

Retaining the FOE, rotation and focal lengths of sets A and B, we generated flow fields for the planar and fractal depth maps with $d = 1.7, 1.5, 1.3$ and 1.1. The high-frequency content of the depth map increases with decreasing d. The results of FOE estimation using our algorithm are shown in Table 4. For both sets A and B, the FOE is estimated with zero error ϵ. The angular error ϵ_ω, indicated in 10^{-3} rad, is very small for A and zero for B. This is not surprising, considering that the true FOE for A lies between grid points. Figs. 5(a) and (b) show the extreme-case depth maps which are planar and fractal respectively, with $d = 1.1$. The corresponding flows generated using sets A and B are plotted in (c) and (d). Finally, the error surface contours, together with true (+) and hypothesized ($\times$) FOE, are shown in Figs. 5(e) and (f) respectively.

5.1.6 Performance vs. Field of View

Until now, we have restricted ourselves to flow fields generated by a camera with a "normal" field of view (FOV), which is typically between 30 and 45 degrees. The nature of the flow field varies considerably as the focal length, or equivalently, the FOV, changes. A robust solution to the 3D SFM problem must operate across a range of fields of view. With noiseless data, our algorithm locates the FOE without error for f between 200 and 800 pixels, corresponding to FOVs between 18 and 65 degrees. To facilitate a better understanding we examine the performance with noise in the flow field. We generated flow fields for sets A and B, with noise level η set to 10.0. This is achieved by choosing appropriate values of σ.

Table 5 provides the results of this experiment. The FOE is estimated with reasonable accuracy, and the rotational error (tabulated in 10^{-3} rad) is very small for all cases. As the focal length increases, so does the σ needed to achieve $\eta = 10.0$. As a consequence, ϵ also shows an upward trend. Fig. 6 shows the extreme cases of our experiment. The synthesized noisy optical flow fields corresponding to $f = 200, 800$ and 800 pixels, using parameter sets A, A, and B, are shown in Figs. 6(a), (c) and (e) respectively. The error contours together with true (+) and estimated ($\times$) FOE are shown in Figs. 6(b), (d) and (f). Despite obvious large perturbations in the flow, our algorithm performs well in locating the FOE.

5.1.7 Effect of Mis-estimated Focal Length

In the above discussion, we have assumed that the focal length of the camera is known accurately. This is realistic in the real world as the physical parameters of the camera are either specified or can be measured using camera calibration algorithms. Nevertheless, characterizing the sensitivity of our algorithm to mis-estimated focal length is useful since this sensitivity determines the deviation of our FOE estimates when the focal length itself is known (or estimated) imprecisely. Furthermore, in situations where the focal length is altogether unknown, this study reveals what parameters, at a minimum, can be computed with some degree of reliability using an arbitrarily chosen focal length.

Our final experiment on synthetic data sets involves estimating the FOE from noiseless, dense flow fields using parameter sets A and B. Unlike the previous experiments, the focal length assumed in the computations is made to vary over the range 200-800 pixels in $2^{0.25}$ factor multiples, while the true focal length is fixed at 400 pixels. Table 6 provides a summary of the results of this experiment. Excluding the extreme wide angle case (200 pixels), the FOE estimate is very good. In the extreme case, the FOE is displaced by 15 and 11 pixels for A and B respectively. These results lead us to claim that the algorithm is relatively insensitive to the focal length, insofar as the FOE estimate is concerned. For the angular velocities, this does not hold. $\hat{\omega}_x$ and $\hat{\omega}_y$ are approximately scaled by the ratio of the true focal length to the chosen focal length. However the rotation along the optical axis is relatively robust to the choice of focal length.

The input optical flow fields for this experiment are shown in Fig. 1(d) and Fig. 2(d). The evolution of the error surface contour with respect to focal lengths of 200, 283, 400, 566 and 800 pixels are shown in Figs. 7(a) and (b), Fig. 3(a), Figs. 7(c) and (d) for the first case, and in Figs. 7(e) and (f), Fig. 3(b), Figs. 7(g) and (h) for the second.

Set A							
Focal Length	$\hat{i}$	$\hat{j}$	ϵ	$\hat{\omega}_x$	$\hat{\omega}_y$	$\hat{\omega}_z$	$\epsilon\omega$
True: 400	102.0	51.0		-2.000	5.000	-8.000	
200	103.5	62.5	11.05	-3.163	9.799	-7.151	4.975
238	103.5	56.5	5.10	-2.972	8.283	-7.396	3.477
283	102.5	53.5	2.00	-2.658	7.025	-7.633	2.157
336	102.5	51.5	0.00	-2.338	5.929	-7.840	1.001
400	101.5	51.5	0.00	-1.985	5.013	-7.995	0.020
476	101.5	50.5	0.00	-1.705	4.216	-8.110	0.845
566	101.5	50.5	0.00	-1.443	3.548	-8.201	1.658
673	101.5	50.5	0.00	-1.219	2.985	-8.265	2.177
800	102.5	50.5	0.00	-1.030	2.498	-8.335	2.704
Set B							
Focal Length	$\hat{i}$	$\hat{j}$	ϵ	$\hat{\omega}_x$	$\hat{\omega}_y$	$\hat{\omega}_z$	$\epsilon\omega$
True: 400	127.5	201.5		5.000	3.000	4.000	
200	132.5	187.5	14.87	8.973	5.745	2.983	4.935
238	131.5	193.5	8.94	7.852	4.880	3.329	3.481
283	130.5	197.5	5.00	6.808	4.142	3.606	2.174
336	128.5	199.5	2.24	5.850	3.543	3.834	1.022
400	127.5	201.5	0.00	5.000	3.000	4.000	0.000
476	126.5	202.5	1.41	4.248	2.540	4.120	0.890
566	125.5	202.5	2.24	3.589	2.152	4.201	1.658
673	124.5	202.5	3.16	3.029	1.824	4.262	2.310
800	124.5	202.5	3.16	2.553	1.534	4.292	2.867

Table 6: Performance as a function of incorrectly estimated focal length

Our experiments using synthetic flow fields illustrate the strengths and limitations of our algorithm, and provide useful insight regarding its domain of applicability. In particular, we have shown that our approach is robust to all the commonly encountered issues in flow field analysis, primary amongst which are noise and sparsity. It remains to be seen how our algorithm can be applied to real-world problems; this is our focus in the next section.

5.2 Experiments on Real Data

Although our technique for FOE estimation works well in theory and in simulations, applying it to solving real world problems presents a whole new set of challenges. In the data processing chain, our algorithm uses a precomputed optical flow field and provides as output the FOE, rotational velocity and depth map (up to a scale factor). Thus, its performance is to some extent circumscribed by the accuracy of the flow field estimation pre-processing stage. While the FOE and rotational velocity estimates are reasonably robust to errors in the flow field, the same is not true of the depth map. This is inevitable, and can be explained by the redundancy. Redundancy, quantified here as the number of equations containing the relevant term, is much higher for the FOE and rotation compared to each inverse depth estimate. Moreover, the depth map is particularly unreliable near the FOE since even small errors in its estimate translate into large relative errors in the coefficient corresponding to the inverse depth.

Thus, for demonstrating practical applicability, we have two problems at hand, *viz.* (1) how to accurately estimate a dense optical flow field and (2) how to effectively use the variably unreliable depth map that our algorithm computes. In this context, it must be mentioned that our algorithm does not enforce a non-negativity constraint on the computed depth map, and negative estimates are very likely to be obtained. In one sense, a negative estimate for depth is preferable to a positive estimate with large error, since the former is obviously invalid while the latter cannot be identified as such.

5.2.1 Applications

As mentioned in the introduction to this paper, 3D SFM has several application areas. We have considered five applications which are, in increasing order of complexity, 3D stabilization, rangefinding or depth estimation, independent motion detection, obstacle detection, and 3D model building for virtual reality. 3D stabilization is the process in which the jerky 3D rotation of a moving camera is compensated by reversing the rotation, in order to stabilize the image. Depth estimation and obstacle detection are self-explanatory. Detection of independently moving foreground objects is trivial when the camera is stationary. However, when the camera is itself moving, this process gets tricky. One cue for detecting such objects is to check for consistency of 3D motion over the scene. Likewise, obstacle detection looks for areas in the image whose depth is inconsistent with a level profile of the ground. 3D model building from a sequence of images is the ultimate application since it extracts all the 3D information in a sequence into a virtual reality model. Re-creating the image sequence is achieved by retracing the 3D path through the model "world". Moreover, alternate paths can be traversed,

generating alternate views that did not exist in the original sequence.

5.2.2 Optical Flow Estimation

Early in our experiments, we noticed that the standard optical flow techniques [32] provided neither the accuracy nor the density needed in most real applications. Despite several decades of work on differential techniques for motion estimation (including ours [33]), the standard algorithms were unable to obtain any reasonable flow field estimate for the real video sequences. The low resolution of the CCD sensor, temporal aliasing caused by coarse sampling intervals, and unsteady motion of the camera are contributing causes to the failure. Barring 3D stabilization (where depth estimates are not required), all the other applications use an exhaustive search based image matching technique developed by us. This process is based on block matching with a small refinement to give subpixel estimates. The flow field used for 3D stabilization is generated using the overlapped basis optical flow field formulation [33]. This is sufficiently accurate and very fast to compute.

There are three functional components to our block matching based optical flow estimation method. The first and most compute-intensive component is the block matching itself. Since this technique is based on matching, it operates on a temporal pair of images. For each pixel in the current image, a 7×7 template is marked around the pixel. This template is compared across a search space in the previous image using the absolute error criterion. The shift corresponding to the minimum total absolute error across the template forms the integral part of the computed flow. We have not used any accelerated search technique here although a multiresolution search can speed up this step substantially.

The second component of our flow computation technique is the estimation of subpixel shift. For this, we use an ad-hoc rule. We compute the total absolute error error at ± 1 pixel from the best shift along both axes. We fit a second-degree polynomial to the error profiles independently in each direction. We pick the minimum of this polynomial, which can be shown to be in $[-0.5, 0.5)$, as the fractional part of the computed flow. The final functional component is the determination of whether or not there is sufficient 2D information at each pixel to reliably compute flow. For this, we use two criteria, *viz.* the determinant and the condition number of the matrix

$$M = \begin{bmatrix} \overline{I_x^2} & \overline{I_x I_y} \\ \overline{I_x I_y} & \overline{I_y^2} \end{bmatrix} \tag{57}$$

where I_x and I_y are the image gradients, and the averages are computed over the 7×7 template. Only when the determinant and condition number are respectively larger and smaller than two preset parameters is the pixel flagged as one for which flow can be computed. In practice, the integral and fractional flow are computed subsequent to this determination.

The subpixel shift estimator is not particularly accurate. However, with no subpixel estimates, the computed depth map often shows a sawtooth pattern. This is somewhat mitigated with the above approach although better techniques like phase correlation will presumably give even better results. One way of improving the subpixel estimates is to blur the input images. An analysis of the histogram of computed velocities shows

a more uniform distribution than with no blur. Undeniably, several possibilities exist for improving the performance of this process. In summary, our flow technique is crude though very effective.

5.2.3 3D Stabilization

Stabilization is a differential process that compensates for the "unwanted" motion in an image sequence. In typical situations, the term "unwanted" denotes the motion in the sequence resulting from the kinematic motion of the camera with respect to an inertial frame of reference. In these situations, the "unwanted" component of motion does not carry any information of relevance to the observer, and indeed strains its functioning. The more common 2D image stabilization techniques apply an interframe translation, similarity, affine or perspective transformation to compensate for image motion. These often perform poorly when the scene is richly structured in 3D.

When a 3D scene is being imaged by an unsteady camera, the resulting image motion is a result of the camera parallax motion (translation) as well as camera rotation. Since the parallax shift cannot be compensated for and is often deliberate or "wanted", it is the rotation that must be anulled. Computing the 3D rotation in an image sequence requires, in effect, that the 3D SFM problem be solved. The rotational component of motion is readily computed once the FOE is determined. For this problem, the depth structure of the scene is largely irrelevant. This allows us to use the overlapped basis technique [33] for computing the flow field with no detriment. The advantages of using the overlapped basis flow field estimator are improved accuracy and computational speed.

Figs. 8 (a) and (b) show the first and hundredth frames of the Martin Marietta sequence. The camera is mounted looking ahead on a vehicle as it traverses unpaved terrain. There is sufficient texture in most of the image, and the interframe displacements are small, permitting differential optical flow computation. The FOE and rotation angles are computed using our algorithm. The estimated pitch, yaw and roll plots are shown in Figs. 8 (c), (d) and (e) respectively. These are in excellent visual compliance with the results obtained by Yao [34].

Fig. 9 demonstrates the effect of 3D stabilization. Fig. 9(a) shows the twentieth frame of the sequence. We chose this frame as it displays higher than average angular deviation from the first frame. With no stabilization, the difference between the twentieth and first frames is shown in Fig. 9(b). The fully stabilized image (compensated for roll, pitch and yaw) and its difference from the first frame are shown in Figs. 9(c) and (d) respectively. In the difference image, areas near the camera show larger deviations than those at a distance. This is the effect of translation of the camera.

Since our algorithm actually computes the three rotation angles for each frame, we can go one step further to perform "selective stabilization". For instance, if we wish to compensate only for camera roll, we disregard the effects of pitch and yaw while derotating the frames. Fig. 9(e) shows the twentieth frame of the Martin Marietta sequence, stabilized for roll only. The difference from the first frame is shown in Fig. 9(f). The parallel horizon and mountain profile in this figure reveals the unstabilized pitch and yaw motion. Extending this concept, one can selectively stabilize for certain frequencies of motion to eliminate handheld jitter while preserving deliberate camera pan, etc.

5.2.4 Independent Motion Detection

Detecting an independently moving foreground object against a stationary background is trivial if the camera is fixed. Frame differencing is often sufficient to accomplish this job. When the camera is moving with respect to the background, more sophisticated techniques must be used. If the background can be assumed to be approximately planar, 2D stabilization steadies the background. The moving foreground can be located by frame differencing the stabilized image sequence.

In a true 3D scenario with the camera undergoing 3D motion and a richly structured 3D background, no global image transformation can stabilize all background objects. Here, we use the consistency of the computed depth map that solves the 3D SFM as a cue to locate independently moving foreground objects. Areas that have a negative depth or very small positive depth are marked as belonging to foreground objects. In theory, this is not a sufficient discriminant. What is actually computed is more accurately the "time-to-collision" and not the inverse depth. In theory, there may exist areas whose time-to-collision with the image plane lies within valid limits. An alternate technique is to compute the difference between the observed optical flow and that calculated using the estimated 3D motion and structure, for each pixel where flow is known. However, in our experiments we found the first cue sufficient.

Fig. 10 shows the results of our first experiment. Two consecutive frames of the sequence, gathered from a forward-translating vehicle on a highway, are shown in Figs. 10(a) and (b). The computed flow between these frames is shown in Fig. 10(c). The depth maps generated from analysis by our algorithm, and after processing, are shown in Figs. 10(d) and (e) respectively. Here, the white areas are those where no depth estimate is available. Background regions are marked in light gray. Receding and approaching areas are indicated by dark gray and black respectively. The raw depth is processed by a series of morphological steps of erosion and dilation. Fig. 10(f) overlays the processed result on the original image. It can be seen that the vehicle near the center of the frame is well segmented as a reading object and the vehicle near the edge is marked as an approaching object.

Our next experiment demonstrates a situation where although the FOE estimation mechanism is ill-suited, the result is very accurate. Figs. 11(a) and (b) are two successive frames of the *Radius* sequence. The camera is mounted looking sideways from a moving vehicle. The camera translates along the image plane and hence the FOE lies at infinity (or very far away from the image center). The central assumptions of our algorithm are violated, rendering our technique inapplicable in principle. The computed flow between frames is shown in Fig. 11(c), revealing the significant rotation as well. Our algorithm estimates the FOE to lie on the right-hand side at the periphery of the search space, which is itself arbitrarily offset to the right by 300 pixels. This indicates the likelihood that the true minimum lies even further beyond. The flow reconstructed from the computed depth map and motion parameters is shown in Fig. 11(d). Visually, this is in excellent agreement with the input flow. The processed depth cue is shown in Fig. 11(e) with the same color legend as used in Fig. 10. The pylon is marked as a distant object and the vehicle is segmented out well (Fig. 11(f)).

5.2.5 Rangefinding

Accuracy in the depth estimates is not very critical to the process of locating moving foreground objects. Accuracy assumes a higher importance for depth estimation or rangefinding. The final applications, obstacle detection and 3D model generation, build on the rangefinding process. Results of our experiments are shown in Figs. 12 and 13. Subfigures (a) and (b) are two consecutive frames of the sequence, (c) is the estimated optical flow and (d) is the computed depth map. White areas indicate no flow and therefore no depth estimate. Darker regions are farther from the camera.

Fig. 12(e) is a plot of the depth as a function of image ordinate. Correcting for projection, Fig. 12(f) plots depth vs. coordinate along the horizontal axis. All rows of the image are collapsed in these plots. Likewise, Fig. 13(e) is a plot of depth vs. image abscissa and (f) corrects for projection. The cylindrical profile of the sponge is seen as the arc formed by the cluster of plots in Fig. 12(f). Similarly, the ground plane shows up as the linear cluster of points in the lower half of Fig. 13(f)

Closer examination of Figs. 12 and 13 shows certain periodicities in plots (e) and (f). This is a result of imperfect subpixel flow estimation. Our observation has been that despite the subpixel correction explained in Section 5.2.2, a histogram of velocities shows a strong preference for integer shifts. This causes a "staircase" effect in the flow estimate which is accentuated in the computed depth map. But, the cluster of row-wise plots of the depth in Fig. 12 (column-wise for Fig. 13) smooths out this artifact.

5.2.6 Obstacle Detection

Building on the rangefinding mechanism, we fit a ground plane to the computed inverse depth map. A plane in 3D shows up as a planar function relating the inverse depth to the image coordinate. Let the ground plane be given by $AX + BY + CZ = 1$. In the image coordinate system,

$$\begin{aligned} Af\frac{X}{Z} + Bf\frac{Y}{Z} + Cf &= f\frac{1}{Z} \\ Ax + By + Cf &= f\frac{1}{Z} \\ ax + by + c &= h(x,y) \end{aligned} \tag{58}$$

which is a planar function for the (scaled) inverse depth $h(x,y)$. We fit a plane to the valid values of the computed inverse depth and look for significant deviations from this plane. While fitting the ground plane, we consider only the lower two-thirds of the image area, assuming, as a rule of thumb, that the top portion of the image looks above the horizon. Some morphological operations are used to clean up the detected regions of interest.

Figs. 14 and 15 show the results of two experiments on obstacle detection. (a) and (b) are consecutive frames for which the flow field is shown in (c). (d) is the computed inverse depth map, where white regions are areas where no flow, and therefore no depth estimate, is available. Darker areas are closer to the camera. (e) is a contour plot of the magnitude of the deviation of the computed inverse depth from the planar fit. The final

results showing detected obstacles superimposed on the original image are shown in (f). False alarms, where present, are small and the segmentation is very good.

5.2.7 3D Model Building

In this final experiment involving real data, we perform an exploratory investigation of the ultimate application of 3D SFM, viz. the process of building 3D models from image sequences. Here, the processing chain does not terminate upon computation of the depth map, or upon locating outliers from the ground plane. Rather, a significant portion of the effort is directed towards digesting the computed depth map values into a meaningful scene model. Even the relatively simple modeling technique used by us in this experiment is highly compute-intensive. Here, more than in the previous examples, the overall accuracy of the process hinges on the pre- and post-processing stages. "Accuracy" is used here as a subjective figure of merit.

Our 3D modeling paradigm is built around the Virtual Reality Modeling Language (VRML). VRML offers a comprehensive vocabulary, ubiquity, and the accompanying visualization tools that allow us to concentrate on building rather than rendering the model. We build our 3D models with only planar faces. Each planar face is bounded by a polygon which is not necessarily convex. Thus, each face is described by its bounding polygonal vertices in 3D, and a superimposed texture map. Building the 3D scene therefore involves breaking up the 2D image into a set of polygonal regions whose internal pixels lie approximately on a plane, followed by computing the orientation and location of each planar region. It can be naively claimed that the latter step can be solved using (58), so what remains is to accurately segment the given scene into polygonal regions. However, there are a few hidden complications that provide daunting challenges at all stages of processing. These steps are described below in detail.

- *Flow Computation and FOE Estimation:* As in our previous experiments, we use two-frame full-search block matching to determine the optical flow. As before, subpixel flow estimates are critical to the overall accuracy. This is followed by estimating the focus of expansion and inverse depth map. The output is a potentially sparse set of inverse depths over the image area.

- *Image Segmentation:* In parallel to the previous step, we segment the image into areas we think fit well to planes in 3D. One possible method is to manually demarcate these segments. Choosing to perform this task automatically, we have developed a simple system to segment the image into areas of almost uniform intensity. Here, we make a critical assumption that adjacent pixels of similar graylevel belong to the same physical plane. This holds reasonably well in real imagery since intensity differences often exist between foreground and background objects. The relative depth difference within each object is nearly zero compared to the absolute depth from the camera. This approach leads us to a paradox. Segmented regions are largely smooth at their interiors and have large derivatives at their boundaries. Often, a boundary is fragmented into numerous tiny regions of no practical value in 3D model building. But, it is at these high-derivative pixels that good flow estimates are available. Thus, useful and reliable flow estimates are available mostly at

and outside the periphery of segmented regions, while most of the interior provides scanty information.

In order to minimize the wastage of useful flow information in fragmented peripheral regions, it is necessary to draw crisp boundaries. Strong edges must be reinforced, and weak edges suppressed, as a precursor to region growing. We use the Graduated Non-Convexity (GNC) algorithm [35] to perform discontinuity-preserving image smoothing. The image intensity data is made to fit to a membrane which is allowed to break under certain stresses. The stiffness of the membrane governs the smoothing, and its yield point governs the ability to preserve discontinuities. The GNC algorithm is iterative, starting with a convex cost function until convergence at the desired error cost function.

The GNC step is followed by region growing. After segmenting the image into near equal intensity regions, the boundaries of these regions are vectorized to form polygons. Segmentation can be improved by using more sophisticated techniques like active contours, or using more informative data like color images.

- *Plane Estimation:* Using the reasoning of (58), we can develop a linear system of equations relating the image plane positions and corresponding scaled inverse depths for points within each region where the depth is known. However, this is an incomplete and logically flawed solution. This is because the computed plane must have positive depth throughout the interior of its corresponding region. A standard linear system of equations does not guarantee this. To better illustrate this point, consider a one-dimensional simplification.

  Let there be ten equally spaced data points $\{d_i, i = 0, 1, \ldots, 9\}$, of which only the first two are known. Let $d_0 = 1$ and $d_1 = 0.8$. With no constraints, the best line fitting these data points does so exactly, and extrapolates $d_i, i = 6, 7, 8, 9$ to be negative. Since this is unacceptable, the non-negativity constraint must be imposed on points where the data is not known. In the 1D case, it is sufficient to impose this constraint at the two endpoints of the data vector. When the data to be fit has a domain in $\Re^2$ and the model is planar, it is sufficient to require that non-negativity be satisfied at the boundary of the region.

  Let $\{x_i, y_i, d_i\}$ be the set of x and y coordinates and corresponding scaled inverse depths at points with known flow, for a particular image region. Also, let $\{\hat{x}_i, \hat{y}_i\}$ be the set of periphery points for the same region. We have the following constrained minimization for solving for the plane parameters (a, b, c):

$$\min \sum_i (ax_i + by_i + c - d_i)^2 \quad \text{s.t.} \quad a\hat{x}_i + b\hat{y}_i + c \geq 0. \tag{59}$$

This is a quadratic programming problem with linear constraints. We include the constraints by forming a composite cost function

$$\begin{aligned} J(\lambda) &= \sum_i (ax_i + by_i + c - d_i)^2 + \lambda \sum_i p\,(a\hat{x}_i + b\hat{y}_i + c) \\ p(x) &= \begin{cases} x^2 & x < 0 \\ 0 & x \geq 0 \end{cases} \end{aligned} \tag{60}$$

with a penalty λ which is gradually increased from zero. Minimizing $J(\lambda)$ is the most computationally expensive step of the process. Faster solutions [37] are commercially available as software packages.

- *Conversion to VRML:* At this stage, we have a list of planar polygons in 3D. The 3D coordinates of their vertices are known and lie ahead of the camera. Two issues remain in converting this list to a usable format. First, the polygons can potentially be non-convex. Moreover, they may be multiply connected (*i.e.* they may have "holes" in them). In either case, the polygons are broken up recursively into triangular faces until only a singly connected convex polygon remains. Together with the triangles, this final polygon forms the 3D model of its parent region.

  The second issue is one of mapping a texture onto each face. Since the normal view of the scene is given, the projection of the texture on each face is known. In order to determine the texture map used to overlay the face, the known projection must be rewarped to the plane of the face. Although this is not mathematically complicated, tricky data storage issues are involved. We have chosen not to carry out this rectification on the texture data, at the cost of enduring visual distortions in our experimental results. In our opinion, this reprojection is better suited to be merged with the rendering mechanism.

- *Experiment:* Using the procedure outlined above, we performed an experiment on the image pair data shown in Figs. 16(a) and (b). The computed optical flow and inverse depth map are shown in Figs. 16(c) and (d) respectively. In the latter, white areas are those where no valid depth estimate exists. Fig. 16(e) shows the segmented regions. Certain areas like the road are oversegmented, *i.e.* several adjacent regions correspond to the same plane. On the other hand, choosing parameters to produce fewer regions leads to undersegmentation, e.g. the van being merged with the sky, which is more undesirable. We solve the quadratic programming step using a conjugate gradient algorithm. The plane parameters (a, b, c) are computed for each region and the inverse depth map reconstructed as shown in Fig. 16(f).

  Fig. 17 shows six rendered views of the 3D model we generated. The first image, Fig. 17(a), is the rendering from the normal viewpoint. "Cracks" in the image are due to the polygonal boundary approximation of regions. Also, tiny fragmented regions are rejected by the algorithm and and are not rendered. Note that the realism of our generated model [4] is better seen using a VRML browser than through printed images. As mentioned earlier, the texture maps are not rectified, giving rise to systematic distortions in the images.

  The first synthetic viewpoint is from above the normal. The ground drops out while the van and other distant objects remain almost at the same level as before (Fig. 17(b)). Next, we generate the views to the left and right of the normal, shown in Figs. 17(c) and (d) respectively. The road surface warps accordingly. Finally, we move the viewpoint ahead of the normal in Figs. 17(e) and (f). The ground diverges outward and the rest of the image changes as expected. This exploratory study

[4] Available at http://www.cfar.umd.edu/ shridhar/Demos/index.html

shows the feasibility of using our algorithm to generate 3D models from video, and also analyzes the post-processing steps, which are indeed more complicated and crucial than depth estimation.

6 Conclusions

The 3D structure from motion problem is very interesting both from the theoretical and the application points of view. Although in theory 3D motion and depth can be recovered simultaneously from a flow field, the solution has proven to be difficult. 3D SFM provides valuable cues for depth estimation, 3D stabilization, robotic navigation, obstacle avoidance, time to collision and virtual reality model generation. But a theoretically sound, robust and computationally tractable solution has eluded researchers. In this paper, we have presented what we believe is a viable solution to the problem.

Our motivation in this work has been to come up with an elegant solution that fully exploits the linear dependence of the optical flow on the focus of expansion and on the scaled inverse depth map. The fundamental result in this paper is a theoretical proof of our claim that a partial search for the focus of expansion is computationally equivalent to performing a finite number [5] of 2D FFTs. Our experimental results on a wide variety of synthetic data representing noise in the flow field, sparsity of the computed flow, uncertainty in the focal length estimate, type of underlying depth map and field of view demonstrates the correctness, robustness and performance envelope of our algorithm. We show, through a variety of experiments, the utility of our algorithm for performing 3D stabilization, rangefinding, independent motion detection, obstacle detection, and 3D virtual reality model building. Our experiments validate the theory behind our approach, and our claims.

3D SFM is an old problem and has received much attention over the years. In parallel, the computational power available to the image analyst has steadily increased over time. At this juncture, it is viable to use fast search-based techniques to solve computer vision problems. An advantage of our algorithm is its ready portability to digital signal processors (DSPs). The powerful FFT support of DSPs makes it conceivable to build an extremely fast (possibly parallel) DSP-based computing engine to implement our algorithm. Having a robust solution to 3D SFM will allow future researchers to concentrate their efforts on higher level vision steps, such as primitive modeling and logical inference, in building computer vision systems.

Acknowledgement

The author would like to thank Profesors Rama Chellappa and Azriel Rosenfeld for their comments and criticisms.

[5] More specifically, 43

Appendix

A Proofs of the Fast Computability Theorems

A.1 Proof of Theorem 1

$$\begin{aligned}
q(i_0, j_0) &= a' \mathcal{S}[i_0, j_0] b \\
&= \sum_{i=0}^{N^2-1} a_i \, b_i \, \mathcal{S}_{i,i}[i_0, j_0] \\
&= \sum_{i=0}^{N^2-1} a_i \, b_i \, \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N}[i_0, j_0] \\
&= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} a_{iN+j} b_{iN+j} \mathbf{S}_{i,j}[i_0, j_0] \\
&= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathbf{C}_{i,j} \mathbf{S}_{i,j}[i_0, j_0] \qquad (61) \\
&= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathbf{C}_{i,j} \mathbf{S}_{(i+i_0) \bmod N, (j+j_0) \bmod N} \qquad (62)
\end{aligned}$$

where the $N \times N$ matrix $\mathbf{C}$ is comprised of component-wise products, $\mathbf{C}_{i,j} = a_{iN+j} b_{iN+j}$. (62) is a 2-D spatial correlation of the "image" $\mathbf{C}$ with a sliding "template" $\mathbf{S}$. Each element of the resultant matrix after correlation is the value of $q(i_0, j_0)$ for the appropriate shift. This operation can be performed in $O(N^2 \log N)$ operations using Fourier domain techniques.

Let $\mathcal{F}$ be the discrete Fourier transform operator. Form the $N \times N$ matrix Q *s.t.* $Q_{i,j} = q(i, j)$. We have

$$\mathcal{F}_Q(k, l) = \mathcal{F}_{\mathbf{C}}(-k, -l) \mathcal{F}_{\mathbf{S}}(k, l) \qquad (63)$$

where the discrete Fourier transforms $\mathcal{F}_Q$, $\mathcal{F}_{\mathbf{C}}$ and $\mathcal{F}_{\mathbf{S}}$ are defined on the matrices $\mathbf{C}$, $\mathbf{S}$ and Q respectively. We can use the Fast Fourier Transform to compute the terms in (63), when $N = 2^k$, in $\mathcal{O}(N^2 \log N)$ steps. The product in Fourier space takes $\mathcal{O}(N^2)$ operations, requiring an overall complexity of $\mathcal{O}(N^2 \log N)$ for the computation of $q(i_0, j_0)$.

A.2 Proof of Theorem 2

Writing out a and b in terms of their components

$$\begin{aligned}
a &= \begin{pmatrix} a_{0,0} & a_{0,1} & a_{1,0} & a_{1,1} & \cdots & a_{N-1,0} & a_{N-1,1} \end{pmatrix}' \\
b &= \begin{pmatrix} b_{0,0} & b_{0,1} & b_{1,0} & b_{1,1} & \cdots & b_{N-1,0} & b_{N-1,1} \end{pmatrix}'
\end{aligned} \qquad (64)$$

and using

$$\mathcal{S}[i_0, j_0] = \text{BlockDiag} \left\{ \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N}[i_0, j_0] \right\} \qquad (65)$$

we get

$$\begin{aligned}
q(i_0,j_0) &= a'\mathcal{S}[i_0,j_0]b \\
&= \sum_{i=0}^{N^2-1} \Big\{ a_{i,0}b_{i,0}\mathbf{S}_{\lfloor i/N\rfloor,i\bmod N_{00}}[i_0,j_0] + a_{i,0}b_{i,1}\mathbf{S}_{\lfloor i/N\rfloor,i\bmod N_{01}}[i_0,j_0] \\
&\quad + a_{i,1}b_{i,0}\mathbf{S}_{\lfloor i/N\rfloor,i\bmod N_{10}}[i_0,j_0] + a_{i,1}b_{i,1}\mathbf{S}_{\lfloor i/N\rfloor,i\bmod N_{11}}[i_0,j_0] \Big\} \\
&= \sum_{i=0}^{N^2-1} \sum_2 \begin{bmatrix} a_{i,0}b_{i,0}\mathbf{S}_{\lfloor i/N\rfloor,i\bmod N_{00}}[i_0,j_0] & a_{i,0}b_{i,1}\mathbf{S}_{\lfloor i/N\rfloor,i\bmod N_{01}}[i_0,j_0] \\ a_{i,1}b_{i,0}\mathbf{S}_{\lfloor i/N\rfloor,i\bmod N_{10}}[i_0,j_0] & a_{i,1}b_{i,1}\mathbf{S}_{\lfloor i/N\rfloor,i\bmod N_{11}}[i_0,j_0] \end{bmatrix} \\
&= \sum_2 \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \begin{bmatrix} \mathbf{C}_{i,j_{00}}\mathbf{S}_{i,j_{00}}[i_0,j_0] & \mathbf{C}_{i,j_{01}}\mathbf{S}_{i,j_{01}}[i_0,j_0] \\ \mathbf{C}_{i,j_{10}}\mathbf{S}_{i,j_{10}}[i_0,j_0] & \mathbf{C}_{i,j_{11}}\mathbf{S}_{i,j_{11}}[i_0,j_0] \end{bmatrix} \\
&= \sum_2 \begin{bmatrix} \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathbf{C}_{i,j_{00}}\mathbf{S}_{i,j_{00}}[i_0,j_0] & \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathbf{C}_{i,j_{01}}\mathbf{S}_{i,j_{01}}[i_0,j_0] \\ \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathbf{C}_{i,j_{10}}\mathbf{S}_{i,j_{10}}[i_0,j_0] & \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathbf{C}_{i,j_{11}}\mathbf{S}_{i,j_{11}}[i_0,j_0] \end{bmatrix} \qquad (66)
\end{aligned}$$

$\sum_2$ denotes the sum of the four entries of the 2×2 matrix summand. $\mathbf{C} \in \Re^{2N\times 2N}$ is formed by the componentwise product of a and b:

$$\mathbf{C}_{i,j} = \begin{pmatrix} \mathbf{C}_{i,j_{00}} & \mathbf{C}_{i,j_{01}} \\ \mathbf{C}_{i,j_{10}} & \mathbf{C}_{i,j_{11}} \end{pmatrix} = \begin{pmatrix} a_{k,0}b_{k,0} & a_{k,0}b_{k,1} \\ a_{k,1}b_{k,0} & a_{k,1}b_{k,1} \end{pmatrix}, \qquad k = iN + j \qquad (67)$$

Each of the entries of the summand matrix in (66) is of the form (61) and is FC according to Theorem 1. The four components of the N^2 signals for each pair (i_0, j_0) can be summed in $O(N^2)$ operations. The overall complexity of evaluating $q(i_0, j_0)$ over the permissible values of i_0 and j_0 is $\mathcal{O}(N^2 \log N)$. Hence, $q(i_0, y_0)$ is FC.

A.3 Proof of Theorem 3

$$\begin{aligned}
q(i_0,j_0) &= a'\mathcal{S}\langle i_0,j_0\rangle b \\
&= \sum_{i=0}^{N^2-1} a_i\, b_i\, \bar{\mathbf{S}}_{\lfloor i/N\rfloor+i_0,i\bmod N+j_0} \\
&= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} a_{iN+j}b_{iN+j}\bar{\mathbf{S}}_{i+i_0,j+j_0} \qquad (68)
\end{aligned}$$

Define $\mathbf{C} \in \Re^{2N\times 2N}$ by

$$\mathbf{C}_{i,j} = \begin{cases} a_{iN+j}b_{iN+j} & i,j \in \{0,1\ldots N-1\} \\ 0 & \text{otherwise.} \end{cases} \qquad (69)$$

Noting that

$$\hat{\mathbf{S}}_{i+i_0,j+j_0} = \hat{\mathbf{S}}_{(i+i_0)\bmod(2N),(j+j_0)\bmod(2N)}, \qquad \forall i,j,i_0,j_0 \in 0,1\ldots N-1 \qquad (70)$$

we get

$$q(i_0,j_0) = \sum_{i=0}^{2N-1}\sum_{j=0}^{2N-1} \mathbf{C}_{i,j}\hat{\mathbf{S}}_{(i+i_0)\bmod(2N),(j+j_0)\bmod(2N)}, \qquad \forall i_0,j_0 \in \{0,1\ldots N-1\} \quad (71)$$

which is of the form (62), with $2N$ replacing N. Thus $q(i_0, j_0)$ can be computed in $\mathcal{O}((2N)^2 \log(2N)) = \mathcal{O}(N^2 \log N)$ operations, proving the theorem. Note that the Fourier technique will calculate $q(i_0, j_0)$ over a larger domain, *viz.* $\{0, 1 \ldots 2N-1\}^2$, of which the useful values are lie within the "North-West" quarter of the matrix.

A.4 Proof of Theorem 4

Writing out a and b in terms of their components as in (64) we get

$$\begin{aligned} q(i_0, j_0) &= a' \mathcal{S}\langle i_0, j_0 \rangle b \\ &= \sum_{i=0}^{N^2-1} \Big\{ a_{i,0}\, b_{i,0}\, [\bar{\mathbf{S}}_{\lfloor i/N \rfloor + i_0, i \bmod N + j_0}]_{00} + a_{i,0}\, b_{i,1}\, [\bar{\mathbf{S}}_{\lfloor i/N \rfloor + i_0, i \bmod N + j_0}]_{01} \\ &\quad + a_{i,1}\, b_{i,0}\, [\bar{\mathbf{S}}_{\lfloor i/N \rfloor + i_0, i \bmod N + j_0}]_{10} + a_{i,1}\, b_{i,1}\, [\bar{\mathbf{S}}_{\lfloor i/N \rfloor + i_0, i \bmod N + j_0}]_{11} \Big\} \\ &= \sum_{2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \begin{bmatrix} a_{iN+j,0} b_{iN+j,0} [\bar{\mathbf{S}}_{i+i_0, j+j_0}]_{00} & a_{iN+j,0} b_{iN+j,1} [\bar{\mathbf{S}}_{i+i_0, j+j_0}]_{01} \\ a_{iN+j,1} b_{iN+j,0} [\bar{\mathbf{S}}_{i+i_0, j+j_0}]_{10} & a_{iN+j,1} b_{iN+j,1} [\bar{\mathbf{S}}_{i+i_0, j+j_0}]_{11} \end{bmatrix} \end{aligned} \tag{72}$$

As in A.3, we define $\mathbf{C} \in \Re^{4N \times 4N}$ by

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{0,0} & \cdots & \mathbf{C}_{0,2N-1} \\ \vdots & \ddots & \\ \mathbf{C}_{2N-1,0} & & \mathbf{C}_{2N-1,2N-1} \end{bmatrix}$$

$$\mathbf{C}_{i,j} = \begin{cases} \begin{bmatrix} a_{iN+j,0} b_{iN+j,0} & a_{iN+j,0} b_{iN+j,1} \\ a_{iN+j,1} b_{iN+j,0} & a_{iN+j,1} b_{iN+j,1} \end{bmatrix} & i, j \in \{0, 1 \ldots N-1\} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \text{otherwise.} \end{cases} \tag{73}$$

Using the reasoning in (70), we get

$$\begin{aligned} q(i_0, j_0) &= \sum_{2} \begin{bmatrix} q_{00} & q_{01} \\ q_{10} & q_{11} \end{bmatrix}, \qquad \forall i_0, j_0 \in \{0, 1 \ldots N-1\} \\ q_{00} &= \sum_{i=0}^{2N-1} \sum_{j=0}^{2N-1} \mathbf{C}_{i,j\,00} [\hat{\mathbf{S}}_{(i+i_0) \bmod (2N), (j+j_0) \bmod (2N)}]_{00} \\ q_{01} &= \sum_{i=0}^{2N-1} \sum_{j=0}^{2N-1} \mathbf{C}_{i,j\,01} [\hat{\mathbf{S}}_{(i+i_0) \bmod (2N), (j+j_0) \bmod (2N)}]_{01} \\ q_{10} &= \sum_{i=0}^{2N-1} \sum_{j=0}^{2N-1} \mathbf{C}_{i,j\,10} [\hat{\mathbf{S}}_{(i+i_0) \bmod (2N), (j+j_0) \bmod (2N)}]_{10} \\ q_{11} &= \sum_{i=0}^{2N-1} \sum_{j=0}^{2N-1} \mathbf{C}_{i,j\,11} [\hat{\mathbf{S}}_{(i+i_0) \bmod (2N), (j+j_0) \bmod (2N)}]_{11} \end{aligned} \tag{74}$$

As before, $\sum_2$ denotes the sum over the four components of the summand matrix. Each of the terms q_{kl} is of the form (71) and is FC by Theorem 3. Since the components themselves can be summed in $\mathcal{O}(N^2)$ operations, the overall complexity in computing $q(i_0, j_0)$ over the domain $\{0, 1 \ldots N-1\}^2$ is $\mathcal{O}(N^2 \log N)$, proving the theorem.

B FastCompute Specification

Objective: To compute $q(i_0, j_0) = a'\mathbf{M}\langle i_0, j_0\rangle b, \forall i_0, j_0 \in \{0, 1 \ldots N-1\}$.
Input: Vectors $a, b \in \Re^{2N^2}$.
Output: Matrix $Q \in \Re^{N \times N}$, where $Q_{i,j} = q(i,j)$.
Internal Data: $\bar{\mathbf{M}}$, given by (42).
Data Structures: Matrices $M_{00}, M_{01}, M_{11} \in \Re^{2N \times 2N}$, initialized to

$$\begin{aligned} M_{00_{i,j}} &= \alpha_{i,j}(j - N + \frac{1}{2})^2 \\ M_{01_{i,j}} &= -\alpha_{i,j}(i - N + \frac{1}{2})(j - N + \frac{1}{2}) \\ M_{11_{i,j}} &= \alpha_{i,j}(i - N + \frac{1}{2})^2 \\ \alpha_{i,j} &= \frac{1}{(i - N + \frac{1}{2})^2 + (j - N + \frac{1}{2})^2} \end{aligned} \tag{75}$$

Matrices $C_{00}, C_{01}, C_{10}, C_{11}, \bar{Q} \in \Re^{2N \times 2N}$, initialized to zero.
Complex valued matrices $\hat{M}_{00}, \hat{M}_{01}, \hat{M}_{11}, \hat{C}_{00}, \hat{C}_{01}, \hat{C}_{10}, \hat{C}_{11}, \hat{Q} \in C^{2N \times 2N}$.

Steps:

1. Initialize data structures as above.
2. Compute the discrete Fourier transforms $\hat{M}_{00}, \hat{M}_{01}$ and $\hat{M}_{11}$ of M_{00}, M_{01} and M_{11} respectively.
3. Write out a and b as in (64).
4. Redefine $C_{00}, C_{01}, C_{10}, C_{11}$ in the upper quarter as

$$\left.\begin{aligned} C_{00_{i,j}} &= a_{iN+j,0} b_{iN+j,0} \\ C_{01_{i,j}} &= a_{iN+j,0} b_{iN+j,1} \\ C_{10_{i,j}} &= a_{iN+j,1} b_{iN+j,0} \\ C_{11_{i,j}} &= a_{iN+j,1} b_{iN+j,1} \end{aligned}\right\} \forall i, j \in \{0, 1 \ldots N-1\} \tag{76}$$

5. Compute the discrete Fourier transforms $\hat{C}_{00}, \hat{C}_{01}, \hat{C}_{10}$ and $\hat{C}_{11}$ of C_{00}, C_{01}, C_{10} and C_{11} respectively.
6. Set $\hat{Q}$ to

$$\begin{aligned} [\hat{Q}]_{k,l} = &[\hat{C}_{00}]^*_{k,l}[\hat{M}_{00}]_{k,l} + [\hat{C}_{01}]^*_{k,l}[\hat{M}_{01}]_{k,l} \\ + &[\hat{C}_{10}]^*_{k,l}[\hat{M}_{01}]_{k,l} + [\hat{C}_{11}]^*_{k,l}[\hat{M}_{11}]_{k,l} \end{aligned} \tag{77}$$

since for a real signal $x(n)$, its Fourier transform $\hat{x}(k)$ displays conjugate symmetry, *i.e.* $\hat{x}(k) = \hat{x}^*(-k)$. Note that this requires complex arithmetic. This summation performs, in effect, the sum over components ($\sum_2$) in A.4 since $\mathcal{F}^{-1}(x) + \mathcal{F}^{-1}(y) = \mathcal{F}^{-1}(x+y)$.

7. Compute the inverse discrete Fourier transform $\bar{Q}$ of $\hat{Q}$.

8. Populate the output matrix Q copying from $\bar{Q}$:

$$Q_{i,j} = \bar{Q}_{i,j}, \qquad i,j \in \{0,1 \ldots N-1\} \tag{78}$$

References

[1] H. Wallach and D. N. O'Connell, The Kinetic Depth Effect, *Journal of Experimental Psychology*, vol. 45, pp. 205-217, 1953.

[2] J. J. Gibson, *The Perception of the Visual World*, Houghton Mifflin, Boston, 1950.

[3] J. J. Gibson, Optical Motions and Transformations as Stimuli for Visual Perception, *Psychological Review*, vol. 64, pp. 288-295, 1957.

[4] S. Ullman, The Interpretation of Structure from Motion, *Proceedings of the Royal Society of London B*, vol. 203, pp. 405-426, 1979.

[5] J. J. Koenderink and A. J. van Doorn, Local Structure of Movement Parallax of the Plane, *Journal of the Optical Society of America A*, vol. 66, pp. 717-723, 1976.

[6] K. Nakayama and J. M. Loomis, Optical Velocity Patterns, Velocity-Sensitive Neurons and Space Perception: A Hypothesis, *Perception*, vol. 3, pp. 63-80, 1974.

[7] H. C. Longuet-Higgins and K. Prazdny, The Interpretation of a Moving Retinal Image, *Proceedings of the Royal Society of London B*, vol. 208, pp. 385-397, 1980.

[8] A. M. Waxman and S. Ullman, Surface Structure and Three Dimensional Motion from Image Flow Kinematics, *International Journal of Robotics Research*, vol. 4, num. 3, pp. 72-94, 1985.

[9] A. R. Bruss and B. K. P. Horn, Passive Navigation, *Computer Vision, Graphics and Image Processing*, vol. 7, pp. 3-20, 1983.

[10] G. Adiv, Determining Three Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, pp. 384-401, 1985.

[11] R. Y. Tsai and T. S. Huang, Estimating Three Dimensional Motion Parameters of a Rigid Planar Patch, *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, pp. 1147-1152, 1981.

[12] X. Zhuang and R. M. Haralick, Rigid Body Motion and the Optic Flow Image, *Proceedings of the First Conference on Artificial Intelligence Applications*, IEEE Computer Society, pp. 366-375, 1984.

[13] X. Zhuang *et.al.*, A Simplified Linear Optic Flow Motion Algorithm, *Computer Vision, Graphics, and Image Processing*, vol. 42, pp. 334-344, 1988.

[14] A. M. Waxman, B. Kamgar-Parsi and M. Subbarao, Closed Form Solutions to Image Flow Equations for 3-D Structure and Motion, *International Joural of Computer Vision*, vol. 1, pp. 239-258, 1987.

[15] A. Mitiche, X. Zhuang and R. M. Haralick, Interpretation of Optical Flow by a Rotation Decoupling, *Proceedings of the IEEE Workshop on Motion*, Miami Beach, FL, pp. 195-200, 1987.

[16] K. Prazdny, Determining the Instantaneous Direction of Motion from Optical Flow Generated by a Curvilinearly Moving Observer, *Computer Vision, Graphics, and Image Processing*, vol. 17, pp. 238-248, 1981.

[17] D. T. Lawton, Processing Translation Motion Sequences, *Computer Vision, Graphics, and Image Processing*, vol. 22, pp. 116-144, 1983.

[18] A. Mitiche, *Computation Analysis of Visual Motion*, Plenum Press, New York, 1994.

[19] J. Weng, T. S. Hwang and N. Ahuja, *Motion and Structure from Image Sequences*, Springer-Verlag, Berlin, 1991.

[20] A. D. Jepson and D. J. Heeger, Linear Subspace Methods for Recovering Translational Direction, *Technical Report RBCV-TR-92-40*, University of Toronto, http://www.cs.utoronto.ca/ jepson/abstracts/dither.html, 1992.

[21] N. C. Gupta and L. N. Kanal, 3-D Motion Estimation from Motion Field, *Artificial Intelligence*, vol. 78, pp. 45-86, 1995.

[22] J. Aloimonos and C. M. Brown, Direct Processing of Curvilinear Sensor Motion from a Sequence of Perspective Images, *Proceedings of the IEEE Workshop on Computer Vision: Representation and Analysis*, Annapolis, MD, pp. 72-77, 1984.

[23] S. Negahdaripour and B. K. P. Horn, Direct Passive Navigation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 168-176, 1987.

[24] B. K. P. Horn and E. J. Weldon, Direct Methods for Recovering Motion, *International Journal of Computer Vision*, vol. 2, pp. 51-76, 1988.

[25] C. Fermuller and Y. Aloimonos, Qualitative Egomotion, *International Journal of Computer Vision*, vol. 15, pp. 7-29, 1995.

[26] C. Fermuller and Y. Aloimonos, On the Geometry of Visual Correspondence, *International Journal of Computer Vision*, vol. 21, pp. 223-247, 1997.

[27] S. Fejes and L. S. Davis, What Can Projection of Flow Fields Tell Us About the Visual Motion, *Proceedings of the International Conference on Computer Vision*, Mumbai, India, pp. 979-986, 1998.

[28] E. P. Simoncelli, Distributed Representation and Analysis of Visual Motion, Ph. D. dissertation, MIT, 1993.

[29] G. Adiv, Inherent Ambiguities in Recovering 3-D Motion and Structure from a Noisy Flow Field, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp.477-489, 1989.

[30] G. S. J. Young and R. Chellappa, Statistical Analysis of Inherent Ambiguities in Recovering 3-D Motion from a Noisy Flow Field, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp.995-1013, 1992.

[31] W. H. Press *et.al.*, *Numerical Recipes in C* (2nd ed.), Cambridge University Press, pp.102-104, 1992.

[32] J. L. Barron, D. J. Fleet and S. S. Beauchemin, Performance of Optical Flow Techniques, *International Journal of Computer Vision*, vol. 12, pp. 43-77, 1994. (Sequences and code available at ftp://ftp.csd.uwo.ca/pub/vision.)

[33] S. Srinivasan and R. Chellappa, "Robust Modeling and Estimation of Optical Flow with Overlapped Basis Functions", CAR-TR-845, University of Maryland, 1996.

[34] Y. S. Yao, "Electronic Stabilization and Feature Tracking in Long Image Sequences", Ph.D. dissertation, CAR-TR-790, University of Maryland, 1996.

[35] A. Blake and A. Zisserman, *Visual Reconstruction*, MIT Press, 1987.

[36] A. L. Ames, D. R. Nadeau and J. L. Moreland, *VRML Sourcebook*, Wiley, 1996.

[37] C. T. Lawrence and A. L. Tits, "Feasible Sequential Quadratic Programming for Finely Discretized Problems from SIP", in R. Reemtsen and J.-J. Ruckmann (eds.): *Semi-Infinite Programming*, Kluwer, 1998.

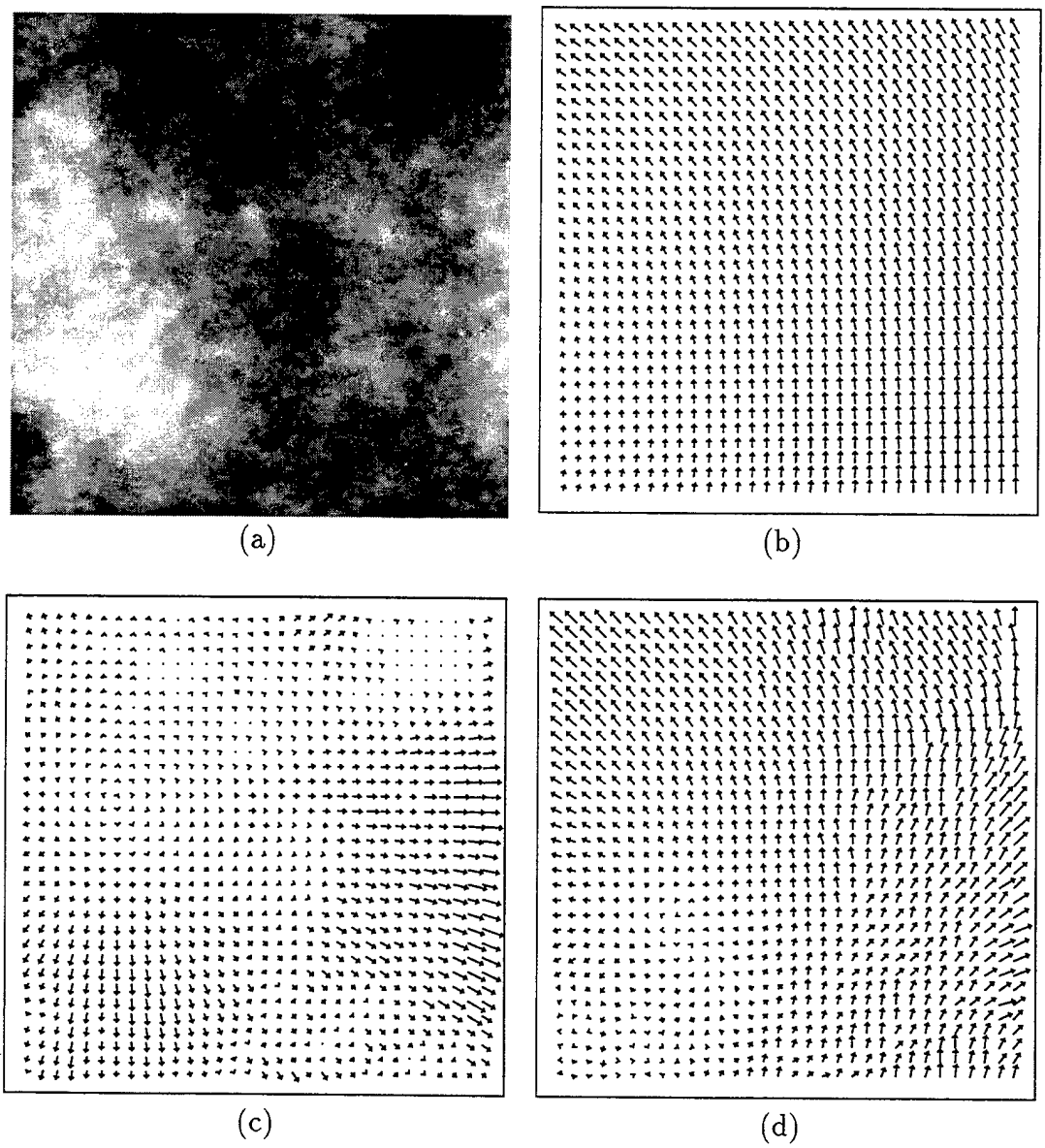


Figure 1: Set A: (a) shows the underlying fractal inverse depth map, (b) the rotational flow, (c) the translational flow and (d) the total optical flow field.

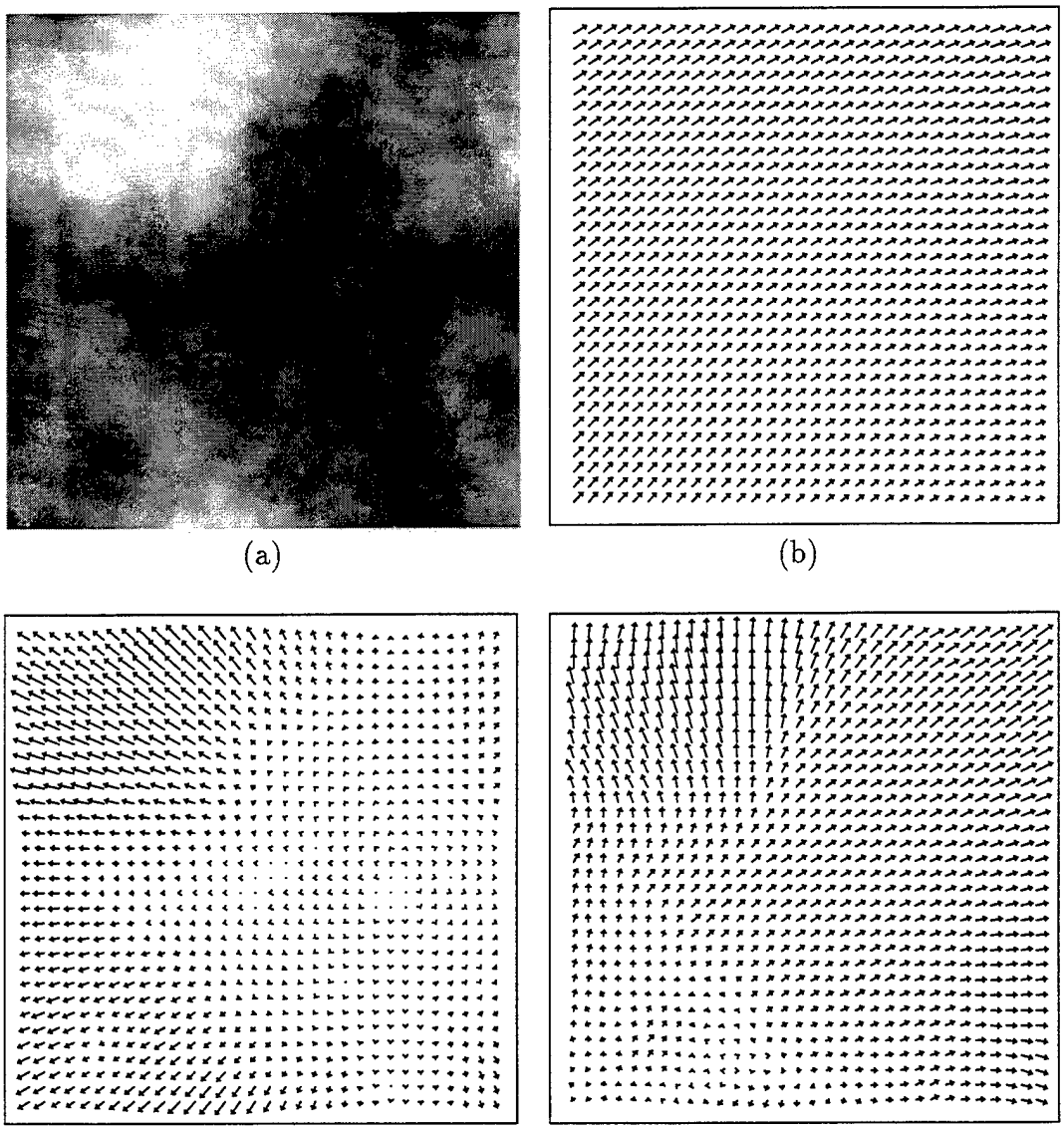


Figure 2: Set B: (a) shows the underlying fractal inverse depth map, (b) the rotational flow, (c) the translational flow and (d) the total optical flow field.

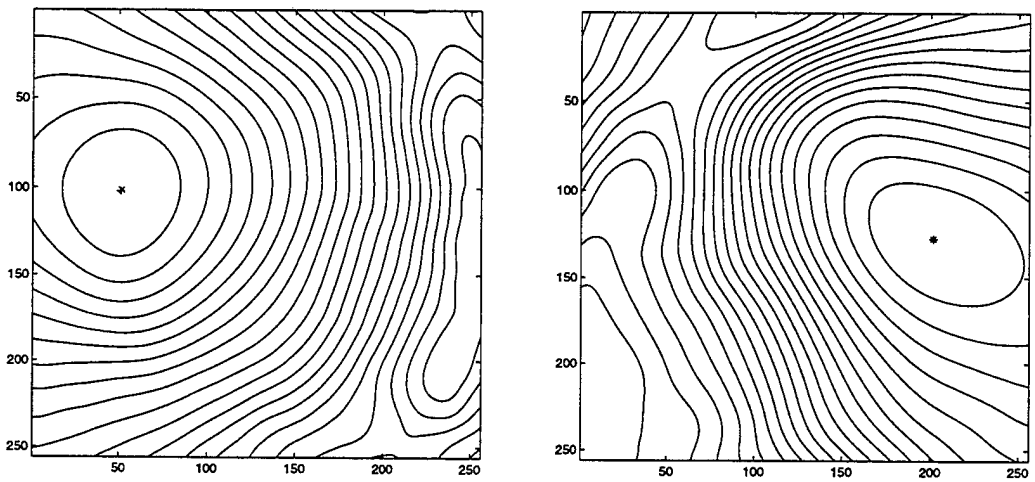


Figure 3: Ideal case performance: (a) and (b) show the error surface contours for sets A and B respectively. The true FOE is marked by a + and the estimated FOE by a $\times$.

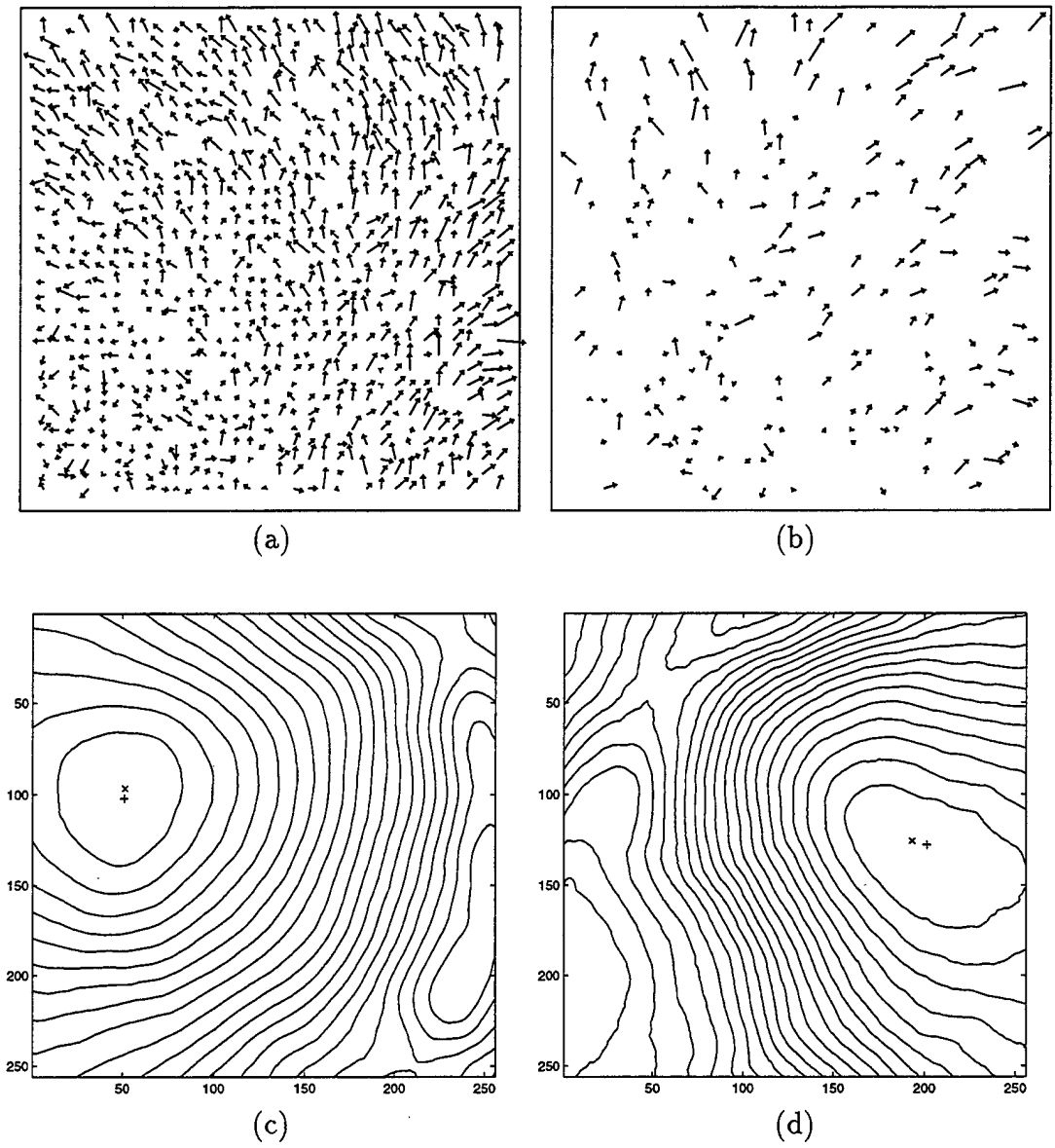


Figure 4: Worst-case performance with a noisy, sparse flow field: (a) *Set A* at 80% density, (b) *Set B* at 20% density, (c) and (d) error surface contours with true FOE (+) and estimated FOE ($\times$) corresponding to (a) and (b) respectively. Zero-mean Gaussian noise with $\sigma = 1.0$ has been added to (a) and (b).

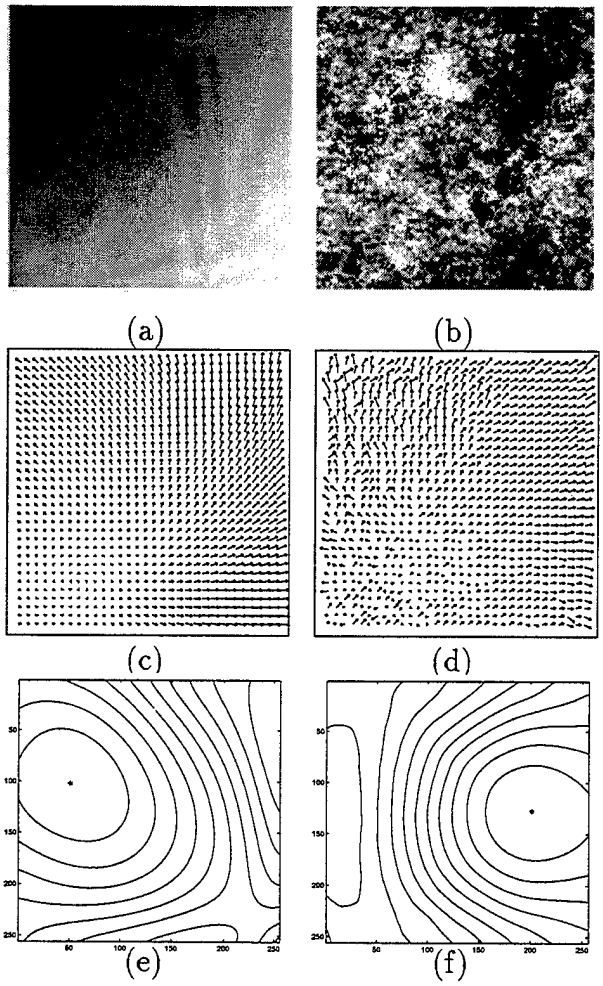


Figure 5: Performance vs. Depth Map Structure: Two extreme depth maps are shown here. (a) shows the *planar* depth map and (b) is a fractal depth map with exponent 1.1. (c) and (d) show flow fields generated with parameter sets A and B and depth maps (a) and (b) respectively. Corresponding error contours, with true and estimated FOEs, are plotted in (e) and (f).

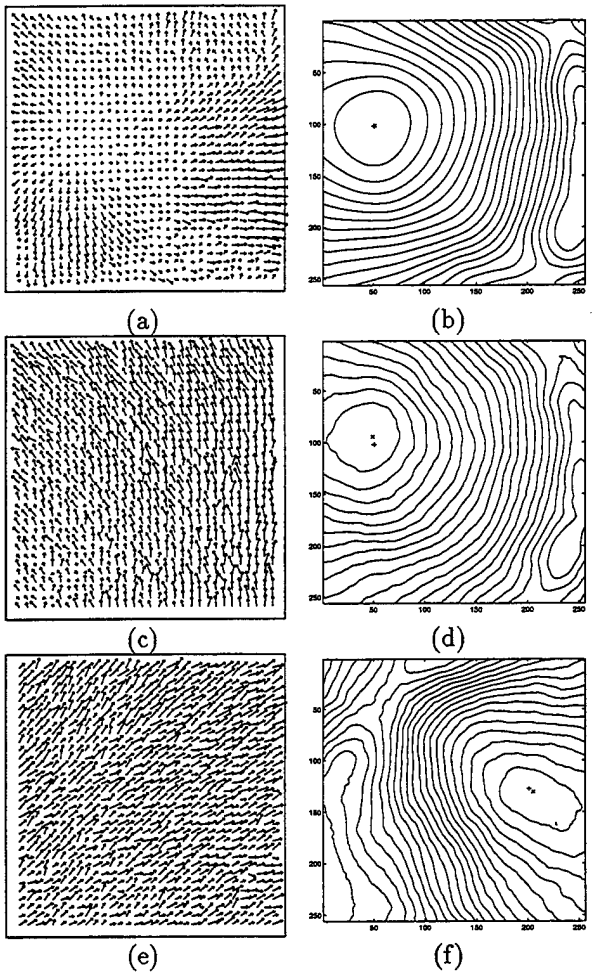


Figure 6: Performance vs. Focal Length: Synthesized noisy flow fields and error surface contours, generated with: (a)-(b) parameter set A, $f = 200$, (c)-(d) A, $f = 800$, and (e)-(f) B, $f = 800$. Noise $\eta = 10.0$ for all cases. The true location is marked by a $+$ and the estimated location by a $\times$.

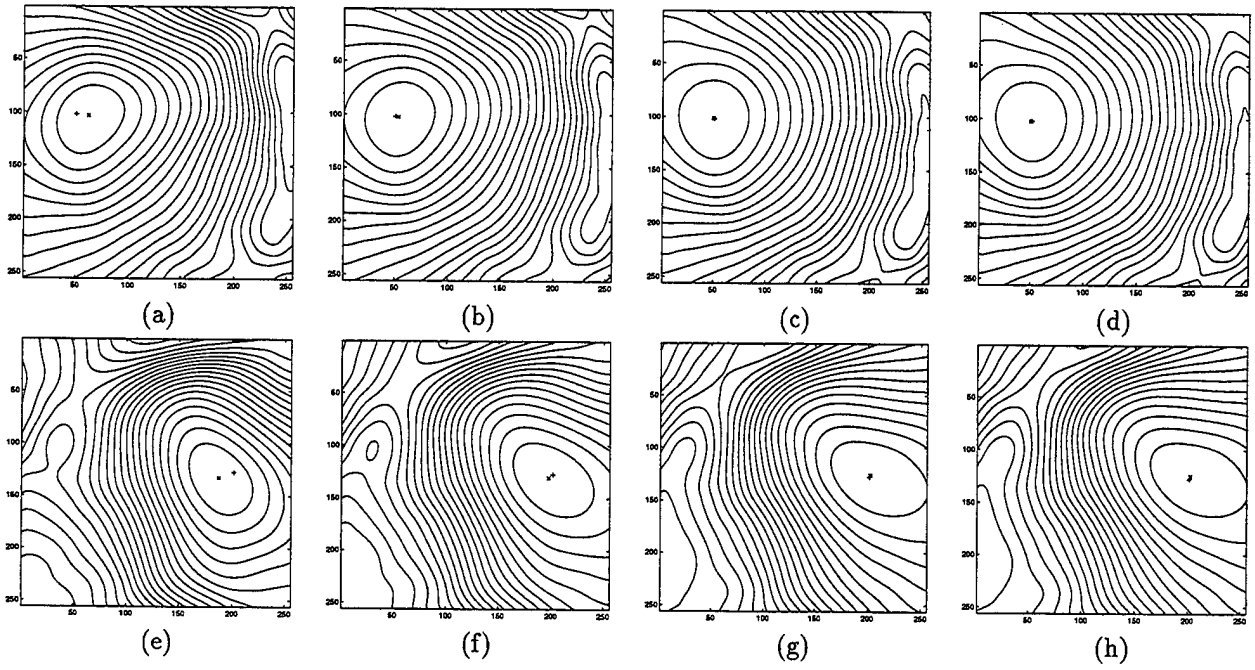


Figure 7: Performance vs. Estimated Focal Length: (a)-(d) error contours for *Set A*, (e)-(h) for *Set B*. The true focal length is 400 pixels. (a)-(d), and (e)-(h) use incorrest estimates of 200, 283, 566 and 800 pixels respectively.

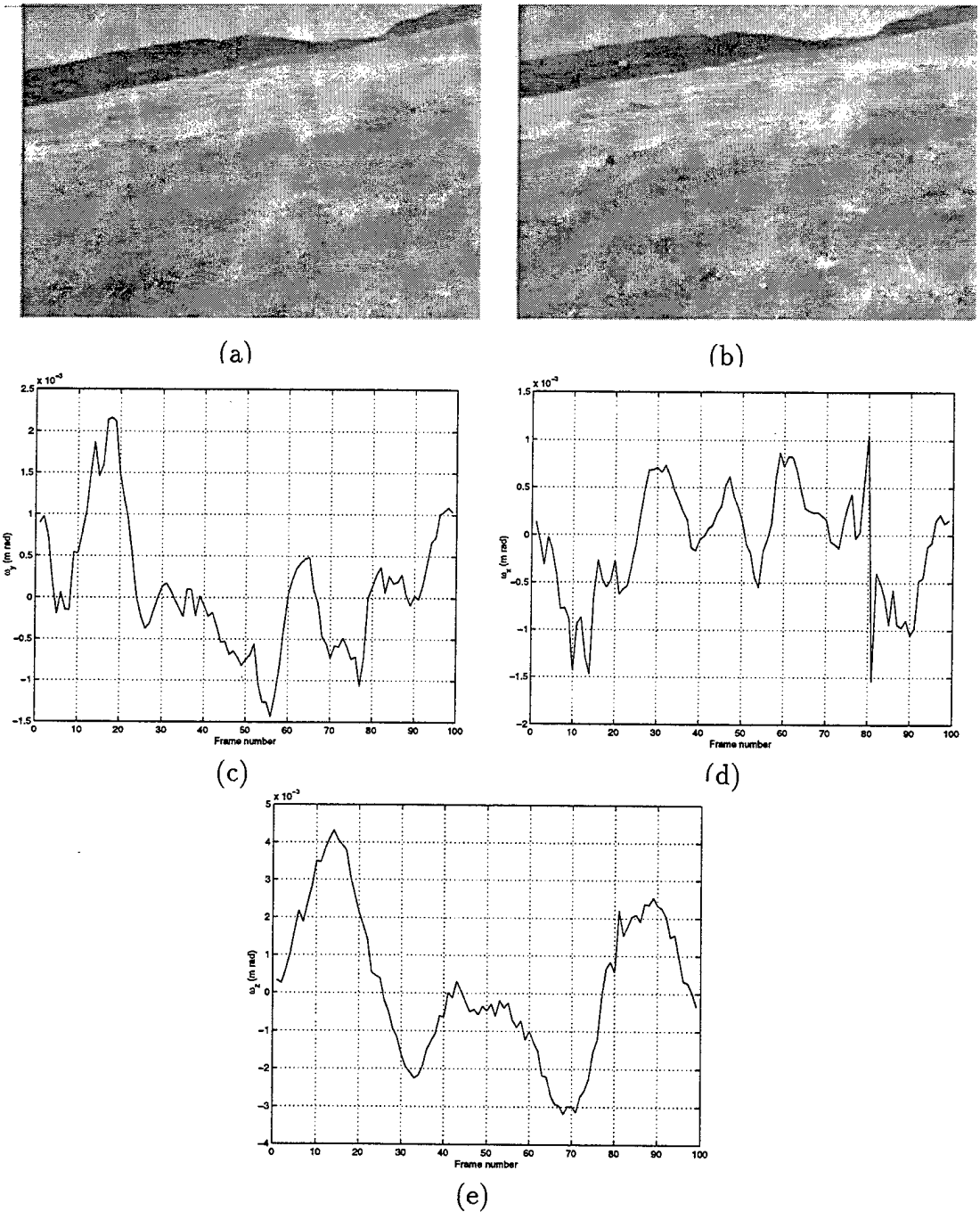


Figure 8: 3D Stabilization: (a) first and (b) hundredth frame of Martin Marietta sequence, (c) pitch, (d) yaw and (e) roll as a function of frame number

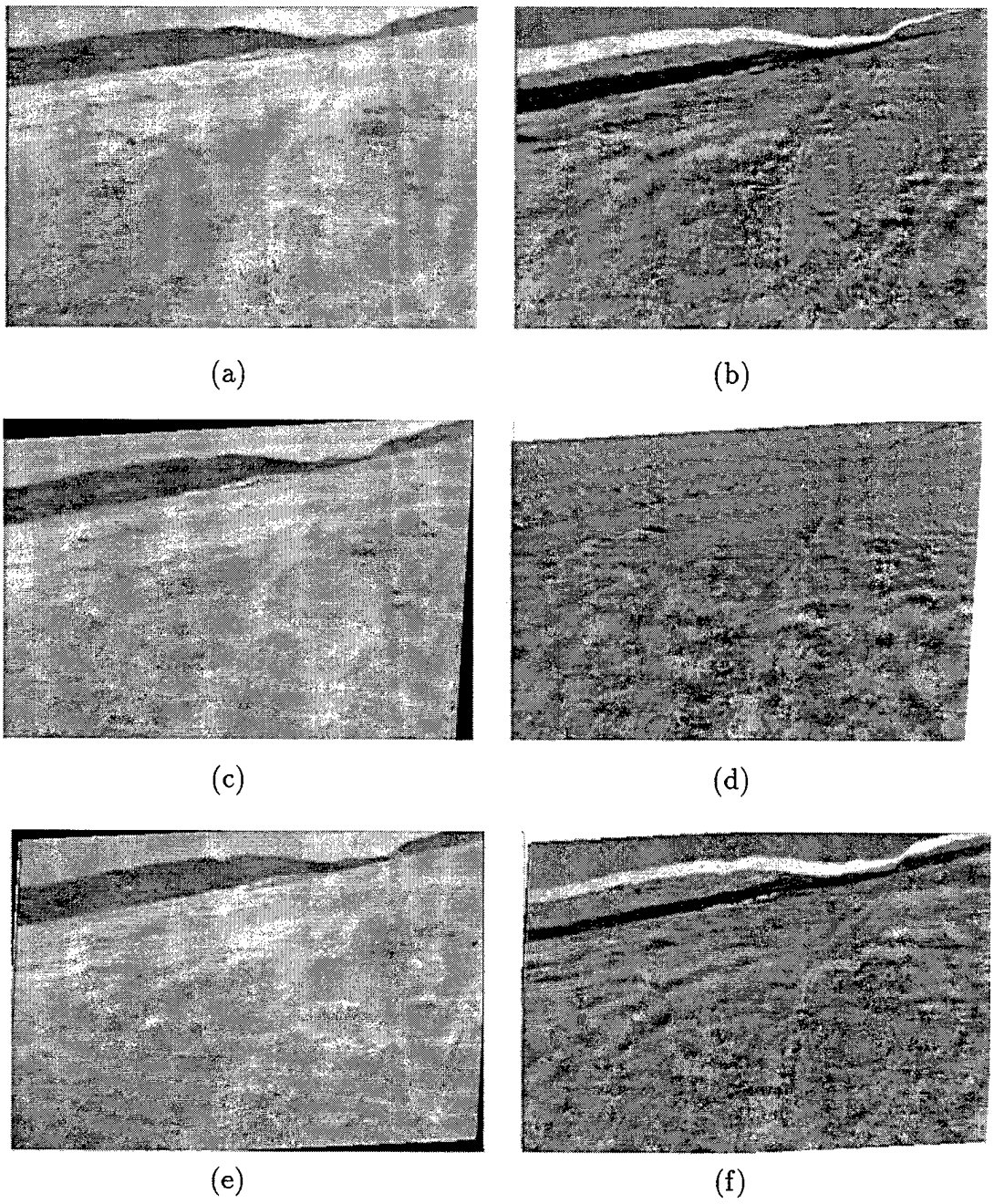


Figure 9: 3D Stabilization: (a) twentieth frame of Martin Marietta sequence, (b) difference between first and twentieth frame with no stabilization, (c) fully stabilized twentieth frame, (d) stabilized difference, (e) stabilized only for roll, (f) difference between roll-stabilized frame and the first frame of the sequence

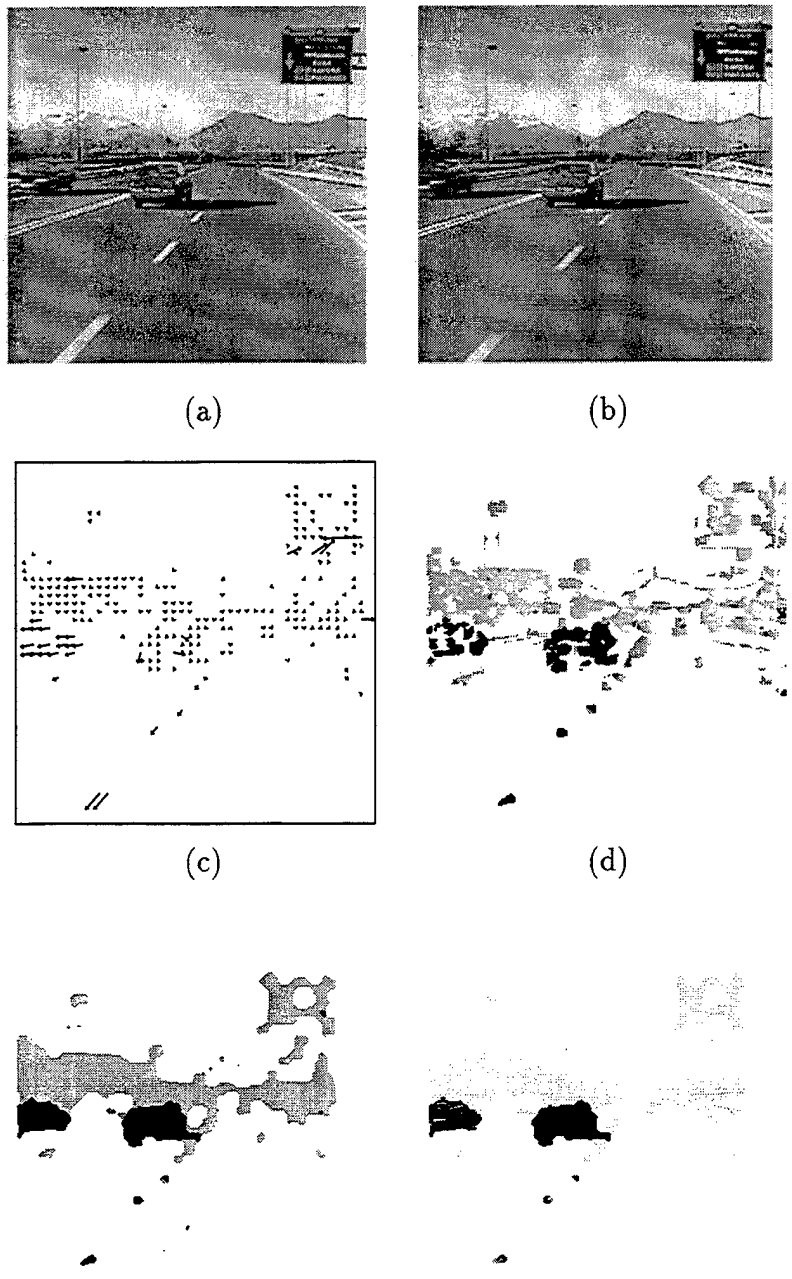


Figure 10: Independent motion detection: (a) and (b) two consecutive frames of the sequence, (c) computed flow, (d) computed depth map, (e) raw and (f) cleaned regions showing independent motion

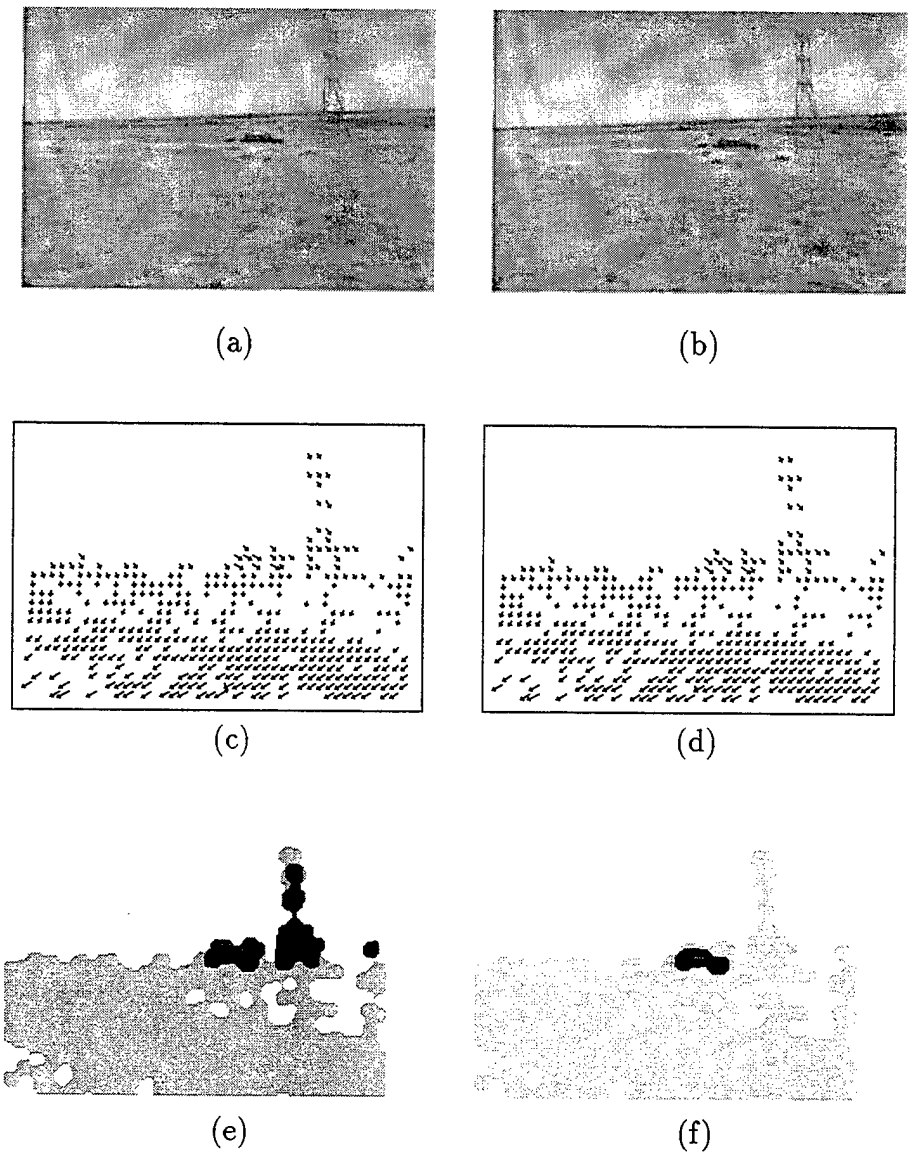


Figure 11: Independent motion detection: (a) and (b) two consecutive frames of the *Radius* sequence, (c) computed flow, (d) reconstructed flow from computed 3D motion parameters and depth map, (e) segmented areas, (f) cleaned regions showing independent motion

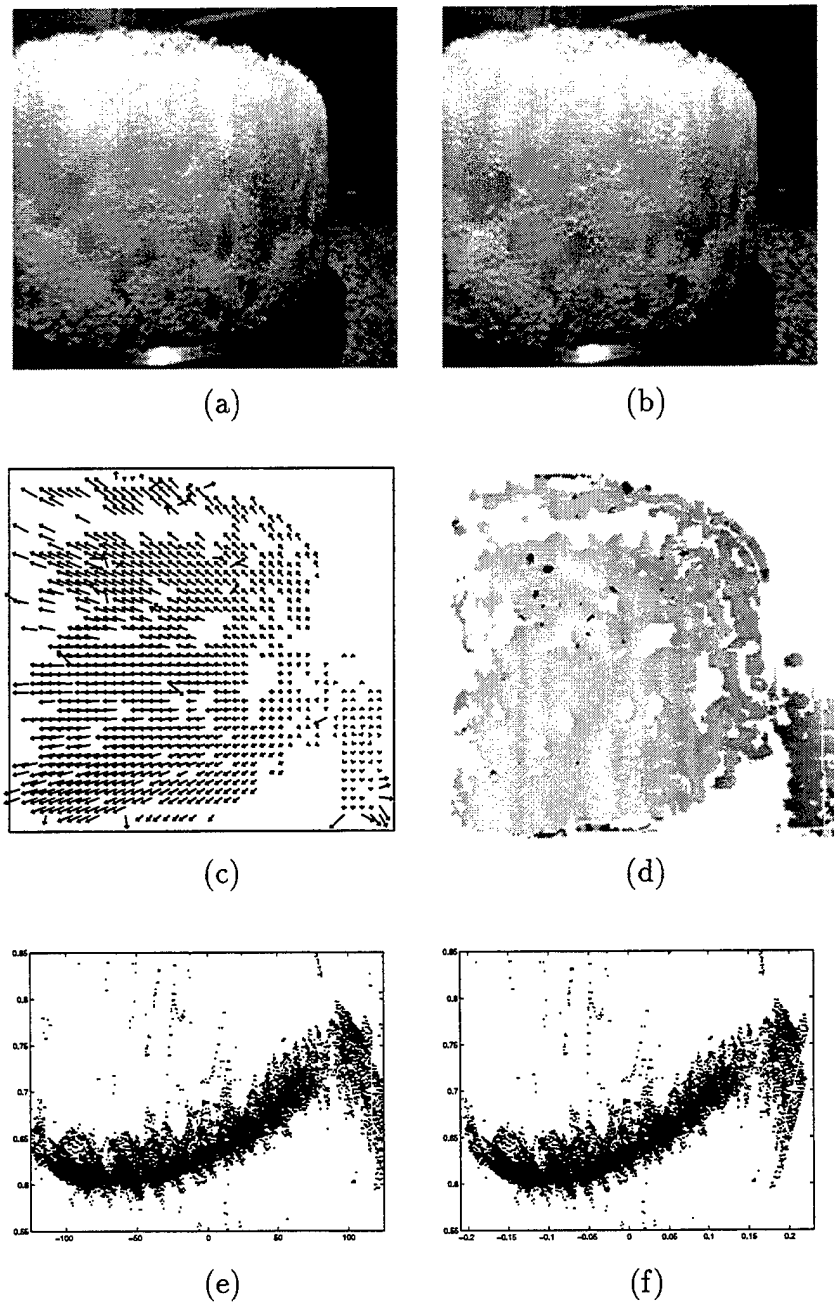


Figure 12: Rangefinding: (a) and (b) two consecutive frames of the sequence, (c) computed flow, (d) computed depth map, (e) plot of depth as a function of image ordinate over the entire height of the image, (f) plot of depth along the horizontal axis; the cylindrical profile of the sponge is evident

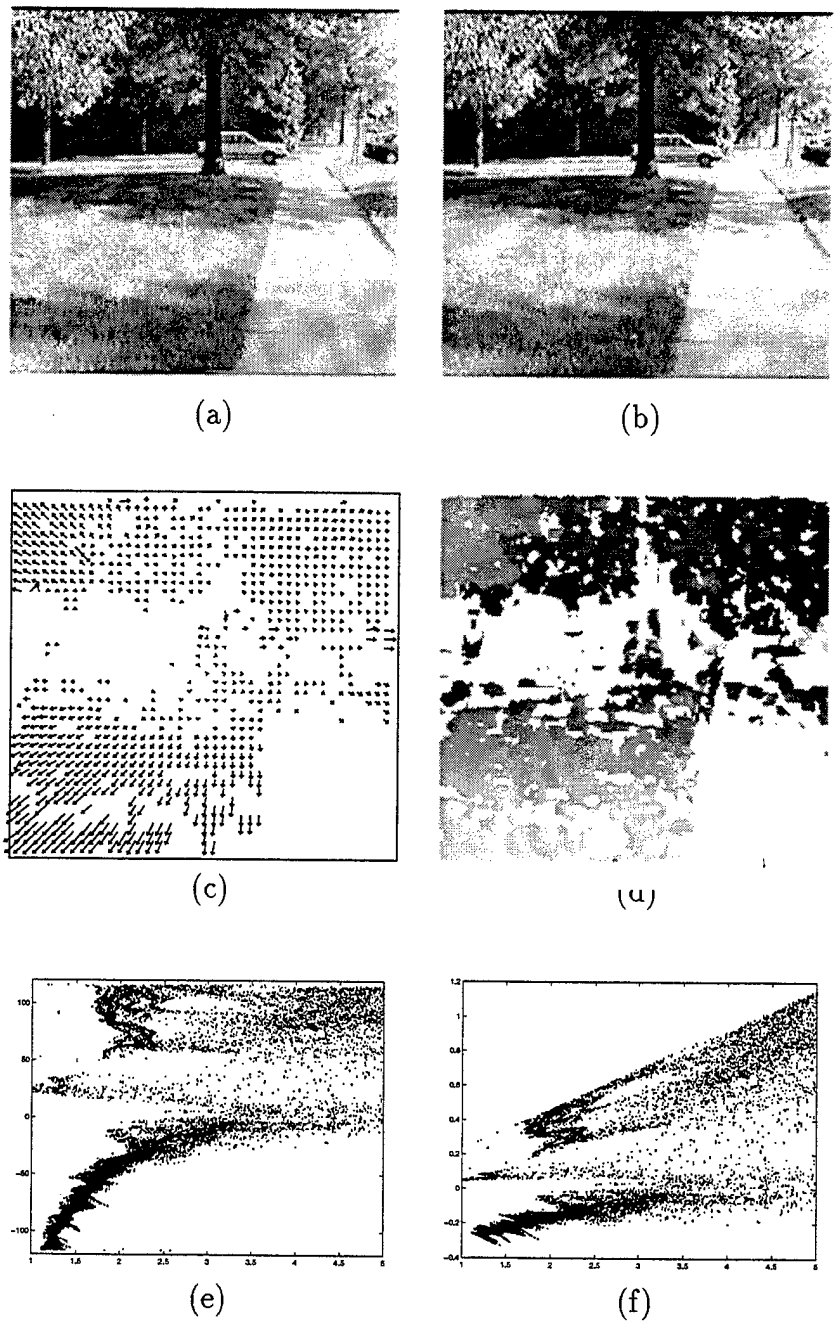


Figure 13: Rangefinding: (a) and (b) two consecutive frames of the sequence, (c) computed flow, (d) computed depth map, (e) plot of depth as a function of image abscissa over the entire width of the image, (f) plot of depth along the vertical axis showing the ground plane as the cluster of points forming a line

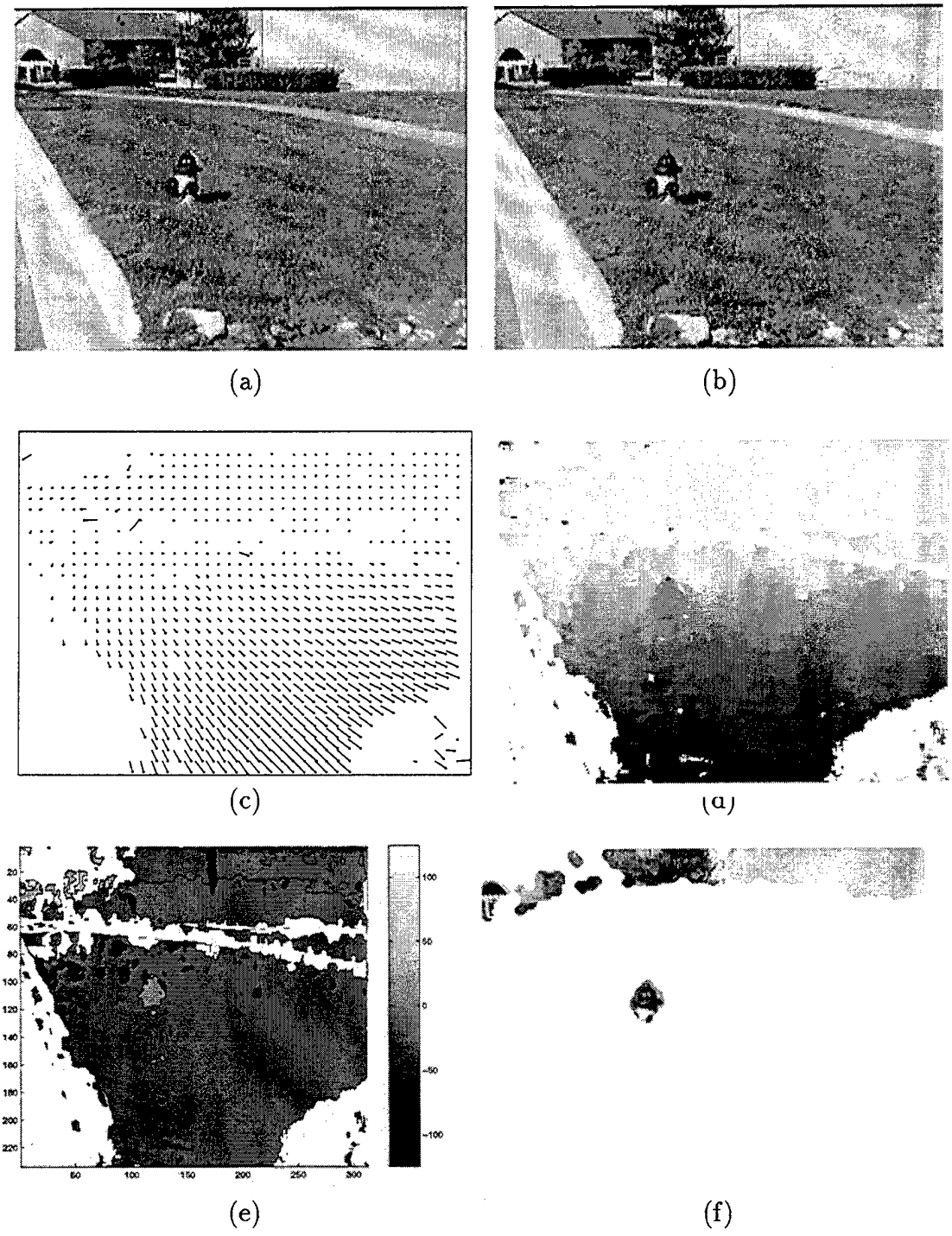


Figure 14: Obstacle detection: (a) and (b) two consecutive frames of the sequence, (c) computed optical flow, (d) inverse depth map, (e) deviation from ground plane, (f) located obstacles

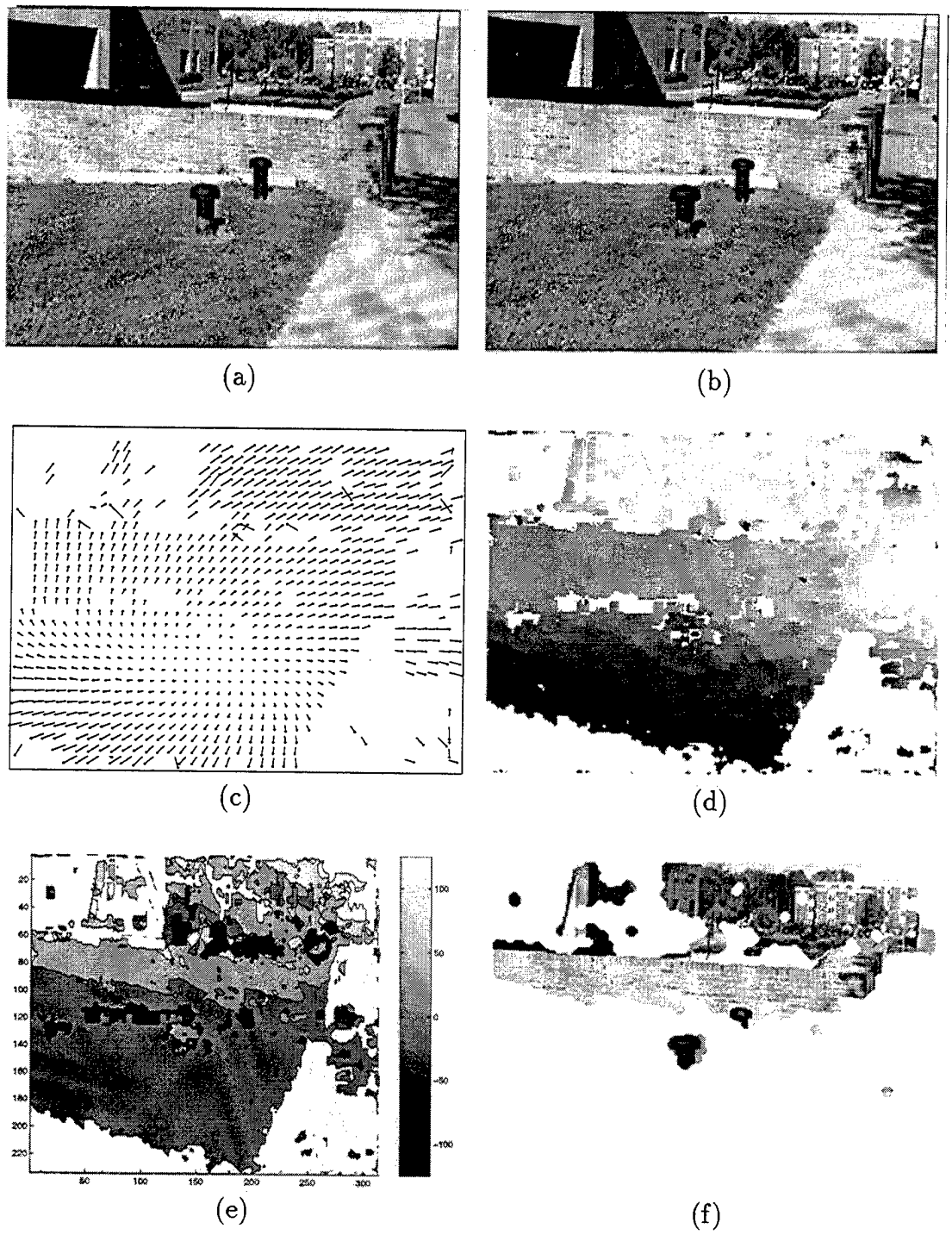


Figure 15: Obstacle detection: (a) and (b) two consecutive frames of the sequence, (c) computed optical flow, (d) inverse depth map, (e) deviation from ground plane, (f) located obstacles

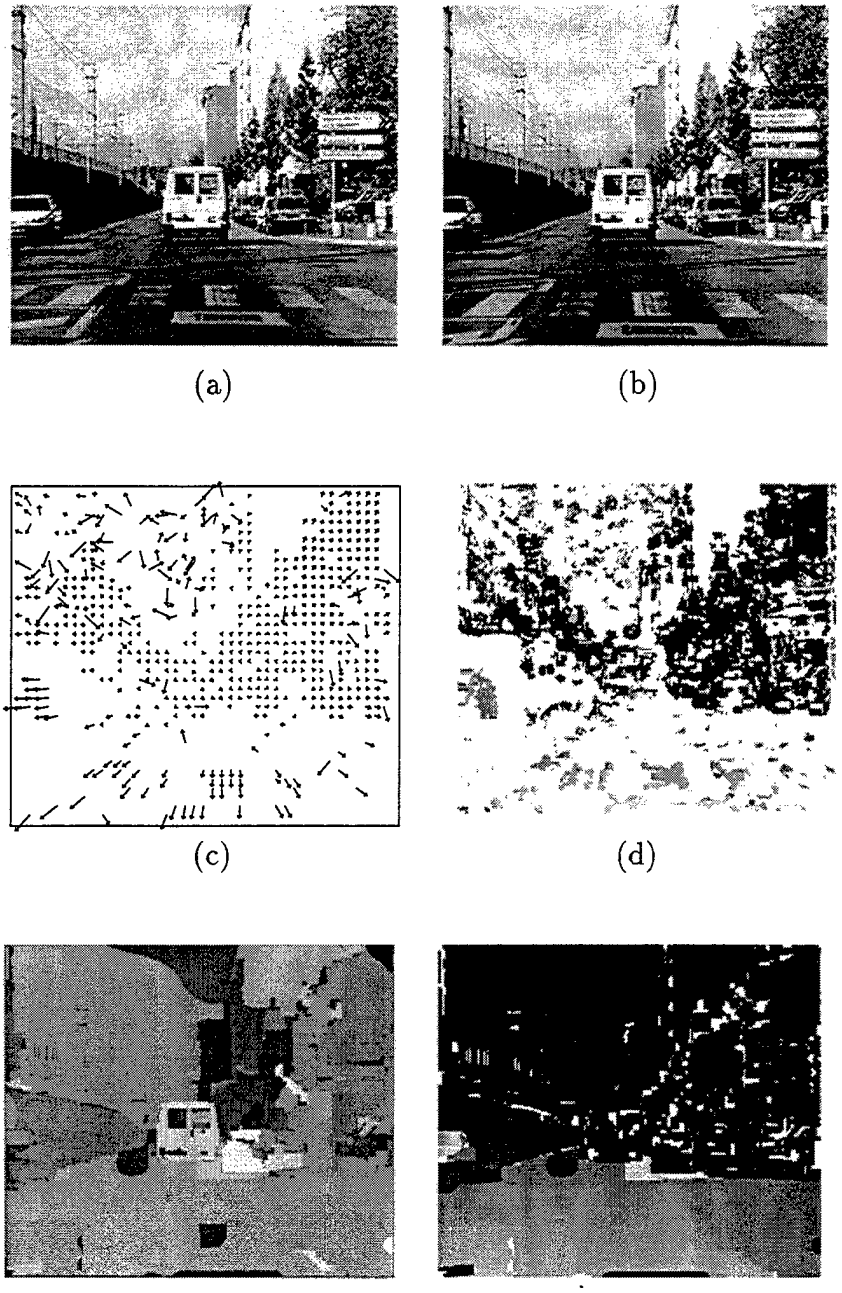
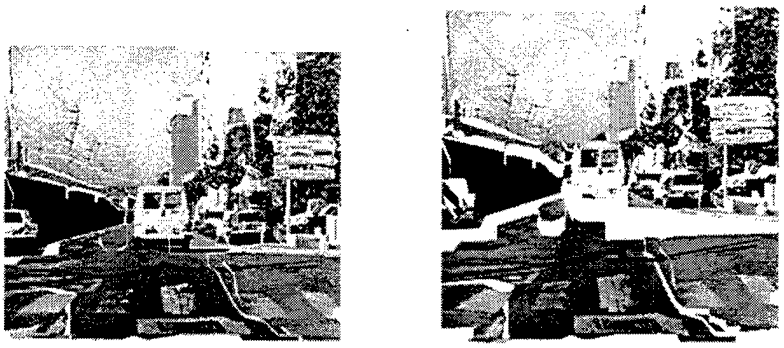
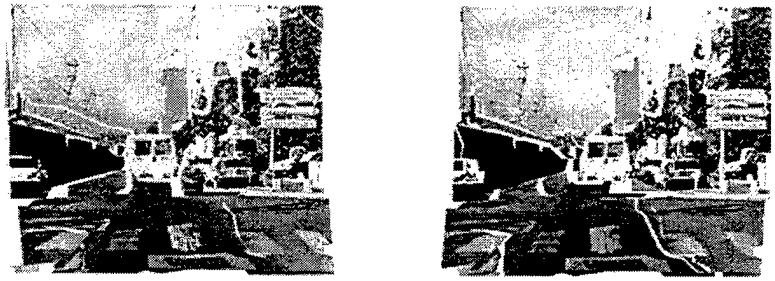


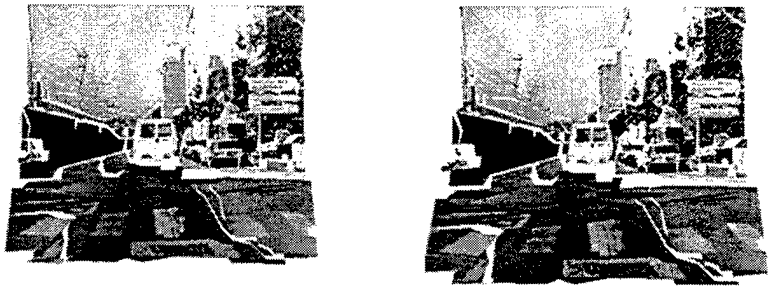
Figure 16: 3D virtual reality model building: (a) and (b) two consecutive frames of the J7 sequence, (c) computed optical flow, (d) inverse depth map, (e) automatically segmented regions, (f) reconstructed inverse depth map



(a) (b)



(c) (d)



(e) (f)

Figure 17: 3D VR model building: (a) reconstructed image from normal viewpoint using 3D model (distortion in the images is caused by the texture map not being orthorectified), (b-f) image generated from other viewpoints: (b) above, (c) to the left, (d) to the right, (e) ahead and (f) further ahead of the normal

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1998	Technical Report

4. TITLE AND SUBTITLE

Extracting Structure from Optical Flow Using the Fast Error Search Technique

5. FUNDING NUMBERS

N00014-95-1-0521

6. AUTHOR(S)

Sridhar Srinivasan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Center for Automation Research
University of Maryland
College Park, MD 20742-3275

8. PERFORMING ORGANIZATION REPORT NUMBER

CAR-TR-893
CS-TR-3923

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 North Quincy Street, Arlington, VA 22217-5660

Advanced Research Projects Agency
3701 North Fairfax Drive, Arlington, VA 22203-1714

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release.
Distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

In this paper, we present a robust and computationally efficient technique for estimating the focus of expansion (FOE) of an optical flow field, using *fast partial search*. For each candidate location on a discrete sampling of the image area, we generate a linear system of equations for determining the remaining unknowns, *viz.* rotation and inverse depth. We compute the least squares error of the system without actually solving the equations, to generate an error surface that describes the goodness of fit across the hypotheses. Using Fourier techniques, we prove that given an $N \times N$ flow field, the FOE can be estimated in $\mathcal{O}(N^2 \log N)$ operations. Since the resulting system is linear, bounded perturbations in the data lead to bounded errors.

We support the theoretical development and proof of our algorithm with experiments on synthetic and real data. Through a series of experiments on synthetic data, we prove the correctness, robustness and operating envelope of our algorithm. We demonstrate the utility of our technique by applying it to the problem areas of 3D stabilization, moving object detection, rangefinding, obstacle detection, and generation of 3D models from video.

14. SUBJECT TERMS

Fast Partial Search, Focus of Expansion, Optical Flow

15. NUMBER OF PAGES

60

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.